RJEŠENJA I PARCIJALNOG ISPITA IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA A)

Zadatak 1 (5 poena)

Ispod je prikazan listing jednog C++ programa, koji ne radi ništa osobito, osim što ispisuje određeni sadržaj na ekran. Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <string>

using namespace std;

void P1(int a, int b) {
  cout << a << b;
  a += 3; b *= 2;
  cout << a << b;
}

void P2(int &a, int &b) {
  cout << a << b;
  a += 3; b *= 2;
  cout << a << b;
}

int F1(int x) {
  return x + 1;
}

int F2(int x) {
  return 2 * x;
}

int main() {
  int a(2), b(5);
  cout << 3 << setw(3) << a << a * a << setw(5) << b * b << endl;
  complex<double> c = 3, i(0, 1);
  cout << c << c * c << " " << c * i << c + c * i << endl;
  P1(b, a);
  cout << a << b << endl;
  P2(b, a);
  cout << a << b << endl;
  string s = "abcdefghijklmnopqrstuvwxyz";
  cout << s.length() << s.size() << s.substr(0, 5) + s.substr(10, 3) << endl;
  int (*npf[5])(int) = {F1, F2, F1, F2, F2};
  int suma(0);
  for(int i = 0; i < 5; i++) suma += (i + 1) * npf[i](i);
  cout << suma;
  return 0;
}
```

```
3▫▫24▫▫▫25
(3,0)(9,0)▫(0,3)(3,3)
528425
528448
2626abcdeklm
78
```

Zadatak 2 (6 poena)

I) Objasnite ukratko šta se podrazumjeva pod pojmom *koncepta* kod generičkih funkcija.

Koncept predstavlja skup ograničenja koji se nameću na neki objekat da bi se on mogao upotrijebiti kao parametar generičke funkcije u određenom kontekstu.

II) Napišite generičku funkciju "`KreirajNiz`" koja prima dva parametra. Prvi parametar predstavlja broj elemenata nekog niza, a drugi parametar je vrijednost kojom treba popuniti elemente niza. Funkcija treba da dinamički kreira niz sa zadanim brojem elemenata, da ga popuni zadanom vrijednošću, i da vrati kao rezultat pokazivač na prvi element novokreiranog niza. U slučaju da kreiranje ne uspije, funkcija treba da baci izuzetak koji se sastoji od teksta "Kreiranje nije uspjelo".

```
template <typename TipElemenata>
TipElemenata *KreirajNiz(int broj_elemenata, TipElemenata popuna) {
  try {
    TipElemenata *dinamicki_niz = new TipElemenata[broj_elemenata];
    for(int i = 0; i < broj_elemenata; i++)
      dinamicki_niz[i] = popuna;
    return dinamicki_niz;
  }
  catch(...) {
    throw "Kreiranje nije uspjelo";
   }
}
```

III) Za funkciju napisanu pod II) odgovorite koji će biti tip elemenata kreiranog niza ako se izvrši svaki od sljedećih poziva:

a) `KreirajNiz(10, 3);`
b) `KreirajNiz<float>(15, 5);`
c) `KreirajNiz(14, 2.);`
d) `KreirajNiz(8, 'a');`
e) `KreirajNiz<int>(27, 'b');`
f) `KreirajNiz(12, "c");`
g) `KreirajNiz<string>(44, "d");`

a) **`int`**
b) **`float`**
c) **`double`**
d) **`char`**
e) **`int`**
f) **`char []`** (ili **`char *`**)
g) `string`

Zadatak 3 (9 poena)

I) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi djelioci broja zadanog kao parametar.

```
vector<int> NadjiDjelioce(int broj) {
  vector<int> djelioci;
  for(int i = 1; i <= broj; i++)
    if(broj % i == 0) djelioci.push_back(i);
  return djelioci;
}
```

II) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu $v_i = (v_i^2 + 1) \bmod 13$, gdje v_i predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.

```
void Transformiraj(vector<int> &v) {
  for(int i = 0; i < v.size(); i++) v[i] = (v[i] * v[i] + 1) % 13;
}
```

III) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta indeks (redni broj) najmanjeg i najvećeg elementa vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "`algorithm`", niti je dozvoljeno sortirati niz.

```
void MinMax(const vector<int> &v, int &indeks_min, int &indeks_max) {
  indeks_min = indeks_max = 0;
  for(int i = 1; i < v.size(); i++) {
    if(v[i] < v[indeks_min]) indeks_min = i;
    if(v[i] > v[indeks_max]) indeks_max = i;
  }
}
```

IV) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima potpunih kvadrata (tj. brojeva koji se mogu napisati kao kvadrati nekog drugog prirodnog broja) ili ne.

```
bool ImaLiPotpunihKvadrata(const vector<int> &v) {
  for(int i = 0; i < v.size(); i++)
    if(int(sqrt(v[i])) == sqrt(v[i])) return true;
  return false;
}
```

V) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.

```
template <typename TipElemenata>
void IspisiBlok(TipElemenata *pocetak, TipElemenata *kraj) {
  while(pocetak != kraj) cout << *pocetak++ << " ";
}
```

VI) Napišite glavni program koji prvo poziva funkciju pod I) da generira vektor svih djelilaca broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod II). Nakon toga, program treba uz pomoć poziva funkcije pod III) da odredi poziciju najmanjeg i najvećeg elementa u vektoru, i da na njihovo mjesto upiše nulu. Konačno, program treba da uz pomoć poziva funkcije pod V) ispiše sve elemente vektora nakon obavljenih manipulacija.

```
int main() {
  int broj;
  cin >> broj;
  vector<int> djelioci = NadjiDjelioce(broj);
  Transformiraj(djelioci);
  int gdje_je_min, gdje_je_max;
  MinMax(djelioci, gdje_je_min, gdje_je_max);
  djelioci[gdje_je_min] = djelioci[gdje_je_max] = 0;
  IspisiBlok(&djelioci[0], &djelioci[djelioci.size()]);
  return 0;
}
```

RJEŠENJA i PARCIJALNOG ISPITA IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA B)

Zadatak 1 (5 poena)

Ispod je prikazan listing jednog C++ programa, koji ne radi ništa osobito, osim što ispisuje određeni sadržaj na ekran. Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <string>

using namespace std;

void P1(int a, int b) {
  cout << a << b;
  a += 2; b *= 3;
  cout << a << b;
}

void P2(int &a, int &b) {
  cout << a << b;
  a += 2; b *= 3;
  cout << a << b;
}

int F1(int x) {
  return x + 1;
}

int F2(int x) {
  return 2 * x;
}

int main() {
  int a(3), b(2);
  cout << 2 << setw(3) << a << a * a << setw(5) << b * b << endl;
  complex<double> c = 2, i(0, 1);
  cout << c << c * c << " " << c * i << c + c * i << endl;
  P1(b, a);
  cout << a << b << endl;
  P2(b, a);
  cout << a << b << endl;
  string s = "01234567890123456789012345678 9";
  cout << s.length() << s.size() << s.substr(0, 4) + s.substr(9, 4) << endl;
  int (*npf[5])(int) = {F1, F2, F2, F1, F2};
  int suma(0);
  for(int i = 0; i < 5; i++) suma += (i + 1) * npf[i](i);
  cout << suma;
  return 0;
}
```

```
2▫▫39▫▫▫▫4
(2,0)(4,0)▫(0,2)(2,2)
234932
234994
303001239012
73
```

Zadatak 2 (6 poena)

I) Objasnite ukratko šta se podrazumjeva pod pojmom *modela nekog koncepta* kod generičkih funkcija.

Model nekog koncepta je bilo koji konkretan tip koji zadovoljava ograničenja koja su nametnuta datim konceptom.

II) Napišite generičku funkciju "`FormirajSekvencu`" koja prima dva parametra. Prvi parametar predstavlja inicijalnu vrijednost elemenata sekvence, dok drugi parametar predstavlja broj elemenata sekvence. Funkcija treba da dinamički kreira niz sa zadanim brojem elemenata, da ga popuni zadanom vrijednošću, i da vrati kao rezultat pokazivač na prvi element novokreiranog niza. U slučaju da kreiranje ne uspije, funkcija treba da baci izuzetak koji se sastoji od teksta "Formiranje sekvence nije obavljeno".

```
template <typename TipElemenata>
TipElemenata *FormirajSekvencu(TipElemenata popuna, int br_elemenata) {
  try {
    TipElemenata *dinamicki_niz = new TipElemenata[br_elemenata];
    for(int i = 0; i < br_elemenata; i++)
      dinamicki_niz[i] = popuna;
    return dinamicki_niz;
  }
  catch(...) {
    throw "Formiranje sekvence nije obavljeno";
   }
}
```

III) Za funkciju napisanu pod II) odgovorite koji će biti tip elemenata kreiranog niza ako se izvrši svaki od sljedećih poziva:

a) `FormirajSekvencu(10, 4);`
b) `FormirajSekvencu<float>(6, 8);`
c) `FormirajSekvencu(3., 9);`
d) `FormirajSekvencu('p', 14);`
e) `FormirajSekvencu<int>('q', 9);`
f) `FormirajSekvencu("r", 12);`
g) `FormirajSekvencu<string>("s", 33);`

a) **`int`**
b) **`float`**
c) **`double`**
d) **`char`**
e) **`int`**
f) **`char []`** (ili **`char *`**)
g) `string`

Zadatak 3 (9 poena)

I) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi prirodni brojevi koji su potpuni kvadrati (tj. koji se mogu napisati kao kvadrat nekog drugog prirodnog broja), a koji su manji od broja zadanog kao parametar.

```
vector<int> NadjiPotpuneKvadrate(int broj) {
  vector<int> potpuni_kvadrati;
  for(int i = 1; i * i < broj; i++) potpuni_kvadrati.push_back(i * i);
  return potpuni_kvadrati;
}
```

II) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu $v_i = (4\,v_i + 7) \bmod 19$, gdje v_i predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.

```
void Transformiraj(vector<int> &v) {
  for(int i = 0; i < v.size(); i++) v[i] = (4 * v[i] + 7) % 19;
}
```

III) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta najmanji i najveći element vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "`algorithm`", niti je dozvoljeno sortirati niz.

```
void MinMax(const vector<int> &v, int &min, int &max) {
  min = max = v[0];
  for(int i = 1; i < v.size(); i++) {
    if(v[i] < min) min = v[i];
    if(v[i] > max) max = v[i];
  }
}
```

IV) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima parnih brojeva ili ne.

```
bool ImaLiParnih(const vector<int> &v) {
  for(int i = 0; i < v.size(); i++)
    if(v[i] % 2 == 0) return true;
  return false;
}
```

V) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.

```
template <typename TipElemenata>
void IspisiBlok(TipElemenata *pocetak, TipElemenata *kraj) {
  while(pocetak != kraj) cout << *pocetak++ << " ";
}
```

VI) Napišite glavni program koji prvo poziva funkciju pod I) da generira vektor svih potpunih kvadrata manjih od broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod II). Nakon toga, program treba uz pomoć poziva funkcije pod III) da odredi i ispiše najmanji i najveći element u vektoru. Konačno, program treba da uz pomoć poziva funkcije pod V) ispiše sve elemente vektora nakon obavljenih manipulacija.

```
int main() {
  int broj;
  cin >> broj;
  vector<int> kvadrati = NadjiPotpuneKvadrate(broj);
  Transformiraj(kvadrati);
  int min, max;
  MinMax(kvadrati, min, max);
  cout << min << " " << max << endl;
  IspisiBlok(&kvadrati[0], &kvadrati[kvadrati.size()]);
  return 0;
}
```

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA C)

Zadatak 1 (5 poena)

Ispod je prikazan listing jednog C++ programa, koji ne radi ništa osobito, osim što ispisuje određeni sadržaj na ekran. Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <string>

using namespace std;

void P1(int a, int b) {
  cout << a << b;
  a += 4; b *= 3;
  cout << a << b;
}

void P2(int &a, int &b) {
  cout << a << b;
  a += 4; b *= 3;
  cout << a << b;
}

int F1(int x) {
  return x + 1;
}

int F2(int x) {
  return 2 * x;
}

int main() {
  int a(4), b(2);
  cout << 3 << setw(3) << a << a * a << setw(5) << b * b << endl;
  complex<double> c = 4, i(0, 1);
  cout << c << c * c << " " << c * i << c + c * i << endl;
  P1(b, a);
  cout << a << b << endl;
  P2(b, a);
  cout << a << b << endl;
  string s = "aabbccddeeffgghhiijjkkllmmnnooppqq";
  cout << s.length() << s.size() << s.substr(2, 5) + s.substr(8, 6) << endl;
  int (*npf[5])(int) = {F2, F1, F1, F1, F2};
  int suma(0);
  for(int i = 0; i < 5; i++) suma += (i + 1) * npf[i](i);
  cout << suma;
  return 0;
}
```

```
3▫▫416▫▫▫▫4
(4,0)(16,0)▫(0,4)(4,4)
2461242
24612126
3434bbccdeeffgg
69
```

Zadatak 2 (6 poena)

I) Objasnite koja je razlika između pojma *tipa* i pojma *koncepta* kod generičkih funkcija.

Koncept je poopćenje pojma tipa, i obuhvata sve tipove koji zadovoljavaju skup ograničenja koje se nameću na tipove koje pripadaju tom konceptu.

II) Napišite generičku funkciju "`Generiraj`" koja prima dva parametra. Prvi parametar predstavlja broj elemenata nekog niza, a drugi parametar je vrijednost kojom treba popuniti elemente niza. Funkcija treba da dinamički kreira niz sa zadanim brojem elemenata, da ga popuni zadanom vrijednošću, i da vrati kao rezultat pokazivač na prvi element novokreiranog niza. U slučaju da kreiranje ne uspije, funkcija treba da baci izuzetak koji se sastoji od teksta "Nema dovoljno memorije".

```
template <typename TipElemenata>
TipElemenata *Generiraj(int broj_elemenata, TipElemenata popuna) {
  try {
    TipElemenata *dinamicki_niz = new TipElemenata[broj_elemenata];
    for(int i = 0; i < broj_elemenata; i++)
      dinamicki_niz[i] = popuna;
    return dinamicki_niz;
  }
  catch(...) {
    throw "Nema dovoljno memorije";
   }
}
```

III) Za funkciju napisanu pod II) odgovorite koji će biti tip elemenata kreiranog niza ako se izvrši svaki od sljedećih poziva:

a) `Generiraj(10, 3);`
b) `Generiraj<float>(15, 5);`
c) `Generiraj(14, 2.);`
d) `Generiraj(8, 'u');`
e) `Generiraj<int>(27, 'u');`
f) `Generiraj(12, "v");`
g) `Generiraj<string>(44, "v");`

a) **`int`**
b) **`float`**
c) **`double`**
d) **`char`**
e) **`int`**
f) **`char []`** (ili **`char *`**)
g) `string`

Zadatak 3 (9 poena)

I) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi stepeni broja 2, a koji su manji od broja zadanog kao parametar.

```
vector<int> NadjiStepeneDvojke(int broj) {
  vector<int> stepeni_dvojke;
  for(int i = 1; i < broj; i *= 2) stepeni_dvojke.push_back(i);
  return stepeni_dvojke;
}
```

II) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu $v_i = (3\,v_i^2 + 2) \bmod 17$, gdje v_i predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.

```
void Transformiraj(vector<int> &v) {
  for(int i = 0; i < v.size(); i++) v[i] = (3 * v[i] * v[i] + 2) % 17;
}
```

III) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta najmanji i najveći element vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "`algorithm`", niti je dozvoljeno sortirati niz.

```
void MinMax(const vector<int> &v, int &min, int &max) {
  min = max = v[0];
  for(int i = 1; i < v.size(); i++) {
    if(v[i] < min) min = v[i];
    if(v[i] > max) max = v[i];
  }
}
```

IV) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima trocifrenih brojeva ili ne.

```
bool ImaLiTrocifrenih(const vector<int> &v) {
  for(int i = 0; i < v.size(); i++)
    if((v[i] >= 100) && (v[i] <= 999)) return true;
  return false;
}
```

V) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.

```
template <typename TipElemenata>
void IspisiBlok(TipElemenata *pocetak, TipElemenata *kraj) {
  while(pocetak != kraj) cout << *pocetak++ << " ";
}
```

VI) Napišite glavni program koji prvo poziva funkciju pod I) da generira vektor svih stepena dvojke manjih od broja unesenog sa tastature, a zatim ga transformira pozivom funkcije pod II). Nakon toga, program treba uz pomoć poziva funkcije pod III) da odredi najmanji i najveći element u vektoru, i da ispiše njihovu razliku. Konačno, program treba da uz pomoć poziva funkcije pod V) ispiše sve elemente vektora nakon obavljenih manipulacija.

```
int main() {
  int broj;
  cin >> broj;
  vector<int> stepeni = NadjiStepeneDvojke(broj);
  Transformiraj(stepeni);
  int min, max;
  MinMax(stepeni, min, max);
  cout << max - min << endl;
  IspisiBlok(&stepeni [0], &stepeni[stepeni.size()]);
  return 0;
}
```

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA D)

Zadatak 1 (5 poena)

Ispod je prikazan listing jednog C++ programa, koji ne radi ništa osobito, osim što ispisuje određeni sadržaj na ekran. Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <string>

using namespace std;

void P1(int a, int b) {
  cout << a << b;
  a += 1; b *= 3;
  cout << a << b;
}

void P2(int &a, int &b) {
  cout << a << b;
  a += 1; b *= 3;
  cout << a << b;
}

int F1(int x) {
  return x + 1;
}

int F2(int x) {
  return 2 * x;
}

int main() {
  int a(6), b(3);
  cout << 1 << setw(3) << a << a * a << setw(5) << b * b << endl;
  complex<double> c = 5, i(0, 1);
  cout << c << c * c << " " << c * i << c + c * i << endl;
  P1(b, a);
  cout << a << b << endl;
  P2(b, a);
  cout << a << b << endl;
  string s = "qwertzuiopasdfghjklyxcvbnm987654321";
  cout << s.length() << s.size() << s.substr(3, 7) + s.substr(14, 4) << endl;
  int (*npf[5])(int) = {F2, F2, F2, F1, F1};
  int suma(0);
  for(int i = 0; i < 5; i++) suma += (i + 1) * npf[i](i);
  cout << suma;
  return 0;
}
```

```
1▫▫636▫▫▫▫9
(5,0)(25,0)▫(0,5)(5,5)
3641863
36418184
3535rtzuiopghjk
57
```

Zadatak 2 (6 poena)

I) Objasnite kakva je razlika između pojma *tipa* i pojma *modela* kod generičkih funkcija.

Modeli su oni tipovi koji zadovoljavaju ograničenja nametnuta nekim konceptom.

II) Napišite generičku funkciju "`AlocirajNiz`" koja prima dva parametra. Prvi parametar predstavlja vrijednost kojoj treba popuniti elemente niza, dok drugi parametar predstavlja broj elemenata niza. Funkcija treba da dinamički kreira niz sa zadanim brojem elemenata, da ga popuni zadanom vrijednošću, i da vrati kao rezultat pokazivač na prvi element novokreiranog niza. U slučaju da kreiranje ne uspije, funkcija treba da baci izuzetak koji se sastoji od teksta "Alokacija nije uspjela".

```
template <typename TipElemenata>
TipElemenata *AlocirajNiz(TipElemenata popuna, int br_elemenata) {
  try {
    TipElemenata *dinamicki_niz = new TipElemenata[br_elemenata];
    for(int i = 0; i < br_elemenata; i++)
      dinamicki_niz[i] = popuna;
    return dinamicki_niz;
  }
  catch(...) {
    throw "Alokacija nije uspjela";
   }
}
```

III) Za funkciju napisanu pod II) odgovorite koji će biti tip elemenata kreiranog niza ako se izvrši svaki od sljedećih poziva:

a) `AlocirajNiz(10, 4);`
b) `AlocirajNiz<float>(6, 8);`
c) `AlocirajNiz(3., 9);`
d) `AlocirajNiz('x', 14);`
e) `AlocirajNiz<int>('y', 9);`
f) `AlocirajNiz("z", 12);`
g) `AlocirajNiz<string>("t", 33);`

a) **`int`**
b) **`float`**
c) **`double`**
d) **`char`**
e) **`int`**
f) **`char []`** (ili **`char *`**)
g) `string`

Zadatak 3 (9 poena)

I) Napišite funkciju koja kao parametar prima jedan prirodan broj, a kao rezultat vraća vektor čiji su elementi svi trocifreni brojevi koji su djeljivi sa brojem koji je zadan kao parametar.

```
vector<int> NadjiTrocifreneDjeljiveSaBrojem(int broj) {
  vector<int> brojevi;
  for(int i = 100; i <= 999; i++)
    if(i % broj == 0) brojevi.push_back(i);
  return brojevi;
}
```

II) Napišite funkciju koja prima kao parametar vektor cijelih brojeva, a koja zatim transformira elemente primljenog vektora po zakonu $v_i = (6\,v_i + 5) \bmod 23$, gdje v_i predstavlja i-ti element vektora, dok "mod" označava operaciju "ostatak pri dijeljenju sa". Funkcija ne vraća nikakav rezultat, već samo modificira vektor koji joj je prenesen kao parametar.

```
void Transformiraj(vector<int> &v) {
  for(int i = 0; i < v.size(); i++) v[i] = (6 * v[i] + 5) % 23;
}
```

III) Napišite funkciju koja kao prvi parametar prima vektor cijelih brojeva, a koja u drugi i treći parametar redom smješta indeks (redni broj) najmanjeg i najvećeg elementa vektora. Za realizaciju ove funkcije nije dozvoljeno koristiti funkcije iz biblioteke "`algorithm`", niti je dozvoljeno sortirati niz.

```
void MinMax(const vector<int> &v, int &indeks_min, int &indeks_max) {
  indeks_min = indeks_max = 0;
  for(int i = 1; i < v.size(); i++) {
    if(v[i] < v[indeks_min]) indeks_min = i;
    if(v[i] > v[indeks_max]) indeks_max = i;
  }
}
```

IV) Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja vraća logičku vrijednost "tačno" ili "netačno", u ovisnosti da li u vektoru ima brojeva koji su stepeni broja 2 ili ne (ukoliko želite, možete koristiti funkciju "`log`" iz bibbioteke "`cmath`", ali to nije neophodno).

```
bool ImaLiStepenaDvojke(const vector<int> &v) {
  for(int i = 0; i < v.size(); i++)
    if(int(log(v[i]) / log(2)) == log(v[i]) / log(2)) return true;
  return false;
}
```

V) Napišite generičku funkciju koja kao parametre prima dva pokazivača na proizvoljni ali isti tip, a koja ispisuje sve elemente bloka koji je omeđen sa ta dva pokazivača. Pri tome se pretpostavlja da prvi pokazivač pokazuje na prvi element bloka, a drugi pokazivač tačno iza posljednjeg elementa bloka.

```
template <typename TipElemenata>
void IspisiBlok(TipElemenata *pocetak, TipElemenata *kraj) {
  while(pocetak != kraj) cout << *pocetak++ << " ";
}
```

VI) Napišite glavni program koji prvo poziva funkciju pod I) da generira vektor svih trocifrenih brojeva djeljivih sa brojem unesenim sa tastature, a zatim ga transformira pozivom funkcije pod II). Nakon toga, program treba uz pomoć poziva funkcije pod III) da odredi poziciju najmanjeg i najvećeg elementa u vektoru, i da ispiše te elemente na ekran (na osnovu određene pozicije). Konačno, program treba da uz pomoć poziva funkcije pod V) ispiše sve elemente vektora nakon obavljenih manipulacija.

```
int main() {
  int broj;
  cin >> broj;
  vector<int> brojevi = NadjiTrocifreneDjeljiveSaBrojem(broj);
  Transformiraj(brojevi);
  int gdje_je_min, gdje_je_max;
  MinMax(brojevi, gdje_je_min, gdje_je_max);
  cout << brojevi[gdje_je_min] << " " << brojevi[gdje_je_max] << endl;
  IspisiBlok(&brojevi[0], &djelioci[brojevi.size()]);
  return 0;
}
```

POPRAVNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (I PARCIJALNI, GRUPA A)

Zadatak 1. (4 poena)

Utvrdite šta će ispisati sljedeći program (odgovor treba biti obrazložen):

```
#include <iostream>
using namespace std;
int &f(int &a, int *b, int c) {
  c *= 20;
  a = *b *a + c;
  int &d = *b;
  a++;
  return d;
}
int main() {
  int a, b, c;
  a = b = c = 30;
  a += ++b;
  f(c, &b, a) = 10;
  cout << a << endl << b << endl << c << endl;
  return 0;
}
```

Rješenje:

```
61
10
2151
```

Zadatak 2. (3 poena)

Napišite funkciju koja prima vektor od n realnih brojeva a_1, a_2, ... a_n kao parametar, i koja računa i vraća kao rezultat vrijednost izraza

$$\sqrt{a_1+\sqrt{a_2+\sqrt{...+\sqrt{a_n}}}}$$

U slučaju da su elementi takvi da rezultat nije realan broj, funkcija treba baciti izuzetak. Vodite računa da u jeziku C++ indeksi vektora idu od 0, a ne od 1!

Rješenje:

```
double KaskadaKorijena(const vector<double> &v) {
  double vrijednost(0);
  for(int i = v.size() - 1; i >= 0; i--) {
    if(v[i] + vrijednost < 0) throw "Negativan broj pod korijenom!\n";
    vrijednost = sqrt(v[i] + vrijednost);
  }
  return vrijednost;
}
```

Zadatak 3. (3 poena)

Napišite funkciju "Sredine" koja treba da ima 3 parametra "N", "AS" i "HS". Funkcija treba da izračuna aritmetičku i harmonijsku sredinu svih cifara u parametru "N" koji je prirodan broj, i da smjesti izračunate vrijednosti u parametre "AS" i "HS" respektivno (podsjetimo se da se harmonijska sredina za n brojeva definira kao recipročna vrijednost aritmetičke sredine njihovih recipročnih vrijednosti). Napišite i mali isječak programa u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

Rješenje:

```
void Sredine(int N, double &AS, double &HS) {
  int broj_cifara(0);
  double suma1(0), suma2(0);
  while(N > 0) {
    broj_cifara++;
    suma1 += N % 10; suma2 += 1. / (N % 10);
    N /= 10;
  }
  AS = suma1 / broj_cifara; HS = broj_cifara / suma2;
}
...
double a_s, h_s;
Sredine(134215,a_s, h_s);
cout << a_s << endl << h_s << endl;
```

Zadatak 4. (3 poena)

Za neki element a_n nekog niza brojeva kaže se da je *lokalni maksimum* tog niza brojeva ukoliko je veći od oba svoja susjeda, tj. ukoliko je $a_n > a_{n-1}$ i $a_n > a_{n+1}$. Napišite generičku funkciju koja prima dva parametra, od kojih je prvi parametar niz elemenata proizvoljnog tipa, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da kao rezultat vrati broj lokalnih maksimuma u tom nizu. Napišite i isječak programa u kojem ćete demonstrirati kako se napisana funkcija može primijeniti na jednom fiksno zadanom nizu od 10 cijelih brojeva.

Rješenje:

```
template <typename UporediviTip>
int BrojLokalnihMaksimuma(UporediviTip niz[], int broj_elemenata) {
  int brojac(0);
  for(int i = 1; i < broj_elemenata - 1; i++)
    if(niz[i] > niz[i - 1] && niz[i] > niz[i + 1]) brojac++;
  return brojac;
}
...
int a[] = {3, 5, 2, 1, 8, 7, 9, 4, 2, 3};
cout << BrojLokalnihMaksimuma(a, 10);
```

Zadatak 5. (3 poena)

Poznato je da se izvod neke funkcije f u tački x može približno izračunati pomoću formule

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

ukoliko je h dovoljno mala vrijednost (tačna vrijednost izvoda dobija se za $h \to 0$). Napišite funkciju "`Izvod`" sa 3 parametra "`f`", "`x`" i "`h`" koja računa približnu vrijednost izvoda funkcije "`f`" u tački "`x`". Pri tome je parametar "`f`" funkcija koja prima realni argument i daje realan rezultat, dok su "`x`" i "`h`" realne vrijednosti. Pri tome, parametar "`h`" se može izostaviti, i tada se uzima podrazumijevana vrijednost $h = 10^{-5}$. Također demonstrirajte kako biste napisanu funkciju "`Izvod`" upotrijebili da izračunate približnu vrijednost izvoda funkcije *sin* u tački $x = 0$.

Rješenje:

```
double Izvod(double f(double), double x, double h = 1e-5) {
  return (f(x + h) - f(x)) / h;
}
...
cout << Izvod(sin, 0);
```

Zadatak 6. (4 poena)

Napišite funkciju sa jednim cjelobrojnim parametrom *n* koja dinamički alocira niz od *n* cijelih brojeva, popunjava ga sa prvih *n* prostih brojeva, i vraća pokazivač na prvi element tako alociranog niza. Ukoliko alokacija ne uspije ili ukoliko je *n* negativan ili nula, funkcija treba baciti izuzetak. Napisanu funkciju demonstrirajte u isječku programa koji ilustrira kako biste mogli pozvati ovu funkciju i osloboditi alociranu memoriju kada ona više nije potrebna. Predvidite i hvatanje eventualno bačenih izuzetaka.

Rješenje:

Predloženo rješenje zasniva se na činjenici da je za testiranje prostosti broja dovoljno ispitati njegovu djeljivost sa prostim brojevima manjim od njega, koji su već nađeni. Jasno je da postoje i drugačija, ali manje efikasna rješenja (postoje i još efikasnija rješenja, ali su ona mnogo složenija):

```
int *AlocirajNizProstih(int broj_elemenata) {
  if(broj_elemenata <= 0) throw "Nekorektan parametar!\n";
  int *niz;
  try {
    niz = new int[broj_elemenata];
  }
  catch(...) {
    throw "Alokacija nije uspjela!\n";
  }
  int brojac(1), tekuci_broj(3);
  niz[0] = 2;
  while(brojac <= broj_elemenata) {
    bool prost(true);
    for(int i = 0; i < brojac; i++)
      if(tekuci_broj % niz[i] == 0) prost = false;
    if(prost) niz[brojac++] = tekuci_broj;
    tekuci_broj += 2;
  }
  return niz;
}
...
try {
  int *niz = AlocirajNizProstih(10);
  for(int i = 0; i < 10; i++) cout << niz[i] << endl;
  delete[] niz;
}
catch(const char poruka[]) {
  cout << poruka;
}
```

POPRAVNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (I PARCIJALNI, GRUPA B)

Zadatak 1. (4 poena)

Utvrdite šta će ispisati sljedeći program (odgovor treba biti obrazložen):

```
#include <iostream>
using namespace std;
int &f(int &a, int *b, int c) {
  c *= 10;
  a = *b *a + c;
  int &d = *b;
  a++;
  return d;
}
int main() {
  int a, b, c;
  a = b = c = 20;
  a += ++b;
  f(c, &b, a) = 30;
  cout << a << endl << b << endl << c << endl;
  return 0;
}
```

Rješenje:

```
41
30
831
```

Zadatak 2. (3 poena)

Napišite funkciju koja prima vektor od n realnih brojeva a_1, a_2, ... a_n kao parametar, i koja računa i vraća kao rezultat vrijednost izraza

$$\frac{1}{a_1}+\frac{1}{a_1+a_2}+\frac{1}{a_1+a_2+a_3}+\ldots+\frac{1}{a_1+a_2+a_3+\ldots+a_n}$$

U slučaju da su elementi takvi da se neki od nazivnika anulira, funkcija treba baciti izuzetak. Vodite računa da u jeziku C++ indeksi vektora idu od 0, a ne od 1!

Rješenje:

```
double SumaRazlomaka(const vector<double> &v) {
  double suma(0), nazivnik(0);
  for(int i = 0; i < v.size(); i++) {
    nazivnik += v[i];
    if(nazivnik == 0) throw "Negativan broj pod korijenom!\n";
    suma += 1 / nazivnik;
  }
  return suma;
}
```

Zadatak 3. (3 poena)

Napišite funkciju "Sredine" koja treba da ima 3 parametra "N", "AS" i "GS". Funkcija treba da izračuna aritmetičku i geometrijsku sredinu svih cifara u parametru "N" koji je prirodan broj, i da smjesti izračunate vrijednosti u parametre "AS" i "GS" respektivno (podsjetimo se da se geometrijska sredina za n brojeva definira kao n-ti korijen iz njihovog produkta). Napišite i mali isječak programa u kojem ćete demonstrirati kako se upotrebljava napisana funkcija.

Rješenje:

```
void Sredine(int N, double &AS, double &GS) {
  int broj_cifara(0);
  double suma(0), produkt(1);
  while(N > 0) {
    broj_cifara++;
    suma += N % 10; produkt *= N % 10;
    N /= 10;
  }
  AS = suma / broj_cifara; GS = pow(produkt, 1. /broj_cifara);
}
...
double a_s, g_s;
Sredine(134215,a_s, g_s);
cout << a_s << endl << g_s << endl;
```

Zadatak 4. (3 poena)

Za neki element a_n nekog niza brojeva kaže se da je *lokalni minimum* tog niza brojeva ukoliko je manji od oba svoja susjeda, tj. ukoliko je $a_n < a_{n-1}$ i $a_n < a_{n+1}$. Napišite generičku funkciju koja prima dva parametra, od kojih je prvi parametar niz elemenata proizvoljnog tipa, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da kao rezultat vrati broj lokalnih minimuma u tom nizu. Napišite i isječak programa u kojem ćete demonstrirati kako se napisana funkcija može primijeniti na jednom fiksno zadanom nizu od 10 realnih brojeva.

Rješenje:

```
template <typename UporediviTip>
int BrojLokalnihMinimuma(UporediviTip niz[], int broj_elemenata) {
  int brojac(0);
  for(int i = 1; i < broj_elemenata - 1; i++)
    if(niz[i] < niz[i - 1] && niz[i] < niz[i + 1]) brojac++;
  return brojac;
}
...
double a[] = {3, 5, 2, 1, 8, 7, 9, 4, 2, 3};
cout << BrojLokalnihMinimuma(a, 10);
```

Zadatak 5. (3 poena)

Poznato je da se izvod neke funkcije *f* u tački *x* može približno izračunati pomoću formule

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

ukoliko je h dovoljno mala vrijednost (tačna vrijednost izvoda dobija se za $h \to 0$). Napišite funkciju "`Izvod`" sa 3 parametra "`f`", "`x`" i "`h`" koja računa približnu vrijednost izvoda funkcije "`f`" u tački "`x`". Pri tome je parametar "`f`" funkcija koja prima realni argument i daje realan rezultat, dok su "`x`" i "`h`" realne vrijednosti. Pri tome, parametar "`h`" se može izostaviti, i tada se uzima podrazumijevana vrijednost $h = 10^{-5}$. Također demonstrirajte kako biste napisanu funkciju "`Izvod`" upotrijebili da izračunate približnu vrijednost izvoda funkcije *sin* u tački $x = 0$.

Rješenje:

```
double Izvod(double f(double), double x, double h = 1e-5) {
  return (f(x + h) - f(x)) / h;
}
...
cout << Izvod(sin, 0);
```

Zadatak 6. (4 poena)

Napišite funkciju sa jednim cjelobrojnim parametrom *n* koja dinamički alocira niz od *n* cijelih brojeva, popunjava ga sa prvih *n* prostih brojeva, i vraća pokazivač na prvi element tako alociranog niza. Ukoliko alokacija ne uspije ili ukoliko je *n* negativan ili nula, funkcija treba baciti izuzetak. Napisanu funkciju demonstrirajte u isječku programa koji ilustrira kako biste mogli pozvati ovu funkciju i osloboditi alociranu memoriju kada ona više nije potrebna. Predvidite i hvatanje eventualno bačenih izuzetaka.

Rješenje:

Predloženo rješenje zasniva se na činjenici da je za testiranje prostosti broja dovoljno ispitati njegovu djeljivost sa prostim brojevima manjim od njega, koji su već nađeni. Jasno je da postoje i drugačija, ali manje efikasna rješenja (postoje i još efikasnija rješenja, ali su ona mnogo složenija):

```
int *AlocirajNizProstih(int broj_elemenata) {
  if(broj_elemenata <= 0) throw "Nekorektan parametar!\n";
  int *niz;
  try {
    niz = new int[broj_elemenata];
  }
  catch(...) {
    throw "Alokacija nije uspjela!\n";
  }
  int brojac(1), tekuci_broj(3);
  niz[0] = 2;
  while(brojac <= broj_elemenata) {
    bool prost(true);
    for(int i = 0; i < brojac; i++)
      if(tekuci_broj % niz[i] == 0) prost = false;
    if(prost) niz[brojac++] = tekuci_broj;
    tekuci_broj += 2;
  }
  return niz;
}
...
try {
  int *niz = AlocirajNizProstih(10);
  for(int i = 0; i < 10; i++) cout << niz[i] << endl;
  delete[] niz;
}
catch(const char poruka[]) {
  cout << poruka;
}
```

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" SA RJEŠENJIMA (GRUPA A)

Napomena: Razumije se da sva prikazana rješenja (osim za Zadatak 1) predstavljaju samo jedno od mnoštva mogućih ispravnih rješenja. Prikazano rješenje je tipično najkraće od svih mogućih rješenja, ali ne uvijek nužno i najefikasnije.

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa, uz kratko obrazloženje zbog čega su rezultati onakvi kakvi jesu. Oprez: bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <string>
#include <cctype>
#include <algorithm>
using namespace std;
void F1(int a, int b) {
  a = b + 2; b = a + 2;
  cout << a << b;
}
void F2(int &a, int &b) {
  a = b + 2; b = a + 2;
  cout << a << b;
}
int *F3(int a, int *b) {
  cout << (*b)++;
  return new int(a);
}
int main() {
  int x(4);
  complex<double> c(2), i(0, 1);
  cout << setw(7) << c * c << c * i << setw(10) << c * i - c << endl;
  string s("qwertzuiopasdfg");
  F1(x, x);
  cout << x << endl;
  F2(x, x);
  cout << x << endl;
  *F3(x, &x) = 10;
  cout << x << endl;
  transform(&s[6], &s[13], &s[2], (int(*)(int))toupper);
  cout << setw(3) << s.length() + s.size() << " " << s << endl;
  return 0;
}
```

Napomena: funkcija "`transform`" transformira blok omeđen sa prva dva parametra tako što na svaki element bloka primjenjuje funkciju zadanu četvrtim parametrom, dok rezultate funkcije smješta u blok čiji je početak određen trećim parametrom. Funkcija "`toupper`" pretvara mala slova u velika.

Rješenje:

```
□□(4,0)(0,2)□□□□(-2,2)
684
888
89
□30□qwUIOPASDpasdfg
```

Zadatak 2 (2,5 poena)

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor u koji su prepisani samo oni elementi primljenog vektora koji su prosti brojevi. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 7, 9, 2, 13, 23, 25, 17 i 91 funkcija treba da vrati kao rezultat vektor čiji su elementi 7, 2, 13, 23 i 17.

Napomena: U svim zadacima gdje nije drugačije rečeno, napišite *samo funkciju*, i ništa više.

Rješenje:

```
vector<int> IzdvojiProste(const vector<int> &v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    bool prost(true);
    for(int j = 2; j < sqrt(v[i]); j++)
      if(v[i] % j == 0) prost = false;
    if(prost) v1.push_back(v[i]);
  }
  return v1;
}
```

Zadatak 3 (2 poena)

Napišite funkciju "`AnalizaStringa`" sa četiri parametra. Prvi parametar je neki dinamički string (tj. objekat tipa "`string`"). Funkcija treba da utvrdi koliko u tom stringu ima znakova koji su velika slova, zatim znakova koji su mala slova, kao i ostalih znakova (tj. znakova koji nisu slova), i da smjesti rezultate analize redom u drugi, treći i četvrti parametar respektivno. Napišite i kratki isječak programa u kojem ćete demonstrirati kako bi se mogla upotrijebiti napisana funkcija.

Rješenje:

```
void AnalizaStringa(const string &s, int &br_v, int &br_m, int &br_o) {
  br_v = br_m = br_o = 0;
  for(int i = 0; i < s.length(); i++)
    if(s[i] >= 'A' && s[i] <= 'Z') br_v++;
    else if(s[i] >= 'a' && s[i] <= 'z') br_m++;
    else br_o++;
}

...
int br_velikih, br_malih, br_ostalih;
AnalizaStringa("sdJHJKsrhUgsZ29rej!!?", br_velikih, br_malih, br_ostalih);
cout << br_velikih << " " << br_malih << " " << br_ostalih;
```

Zadatak 4 (1,5 poen)

Napišite funkciju sa jednim cjelobrojnim parametrom koja kao rezultat vraća najmanju cifru tog broja. Na primjer, ukoliko se funkciji proslijedi broj 35942, funkcija treba da kao rezultat vrati broj 2.

Rješenje:

```
int NajmanjaCifra(int broj) {
  int najmanja(broj % 10);
  while(broj > 0) {
    if(broj % 10 < najmanja) najmanja = broj % 10;
    broj /= 10;
  }
  return najmanja;
}
```

Zadatak 5 (2 poena)

Napišite generičku funkciju sa 3 parametra *x*, *f* i *n*. *x* je vrijednost nekog nedefiniranog tipa T, *f* je funkcija koja prima parametar tipa T i vraća rezultat tipa T, dok je *n* cijeli broj. Funkcija treba da kao rezultat vrati vrijednost $f(f(f(...(f(x))...))$ gdje se funkcija *f* uzastopno primjenjuje *n* puta, odnosno vrijednost koja se dobije kada se na argument *x* funkcija *f* primijeni *n* puta.

Rješenje:

```
template <typename T>
T Kaskada(T x, T f(T), int n) {
  for(int i = 1; i <= n; i++) x = f(x);
  return x;
}
```

Zadatak 6 (2 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`equal`". Ova funkcija vraća kao rezultat logičku logičku vrijednost "**`true`**" ukoliko je blok elemenata između pokazivača *p1* i *p2* identičan po sadržaju bloku elemenata na koji pokazuje pokazivač *p3*, a u suprotnom vraća logičku vrijednost "**`false`**". Napišite sami generičku funkciju "`JednakiBlokovi`" koja radi posve istu stvar kao i funkcija "`equal`".

Rješenje:

```
template <typename Pokazivac>
bool JednakiBlokovi(Pokazivac p1, Pokazivac p2, Pokazivac p3) {
  while(p1 != p2)
    if(*p1++ != *p3++) return false;
  return true;
}
```

Zadatak 7 (2 poena)

Pretpostavimo da želimo sortirati niz kompleksnih brojeva tako da kompleksni broj z_1 dolazi ispred kompleksnog broja z_2 ako i samo ako je imaginarni dio od z_1 manji od imaginarnog dijela od z_2, ili ako su imaginarni dijelovi od z_1 i z_2 jednaki, ali je realni dio od z_1 manji od realnog dijela od z_2. Definirajte odgovarajuću funkciju kriterija, i napišite isječak programa u kojem ćete pokazati kako biste deklarirali niz kompleksnih brojeva, napunili ga vrijednostima unesenim sa tastature i sortirali ga u traženi poredak korištenjem funkcije "`sort`" iz biblioteke "`algorithm`".

Napomena: Funkcija "`sort`" prima kao parametre dva pokazivača koji omeđuju blok, i opcionalno, funkciju kriterija kao treći parametar. Realni i imaginarni dio kompleksnog broja mogu se dobiti pozivom funkcija "`real`" i "`imag`" respektivno (prosljeđujući im kompleksni broj kao parametar).

Rješenje:

```
bool Kriterij(complex<double> z1, complex<double> z2) {
  return imag(z1) < imag(z2) ||
    imag(z1) == imag(z2) && real(z1) < real(z2);
}

...
complex<double> niz[100];
for(int i = 0; i <= 100; i++) cin >> niz[i];
sort(niz, niz + 100, Kriterij);
```

Zadatak 8 (3 poena)

Napišite generičku funkciju koja prima kao parametar matricu organiziranu kao vektor vektora, čiji su elementi proizvoljnog tipa. Funkcija treba da izvrši dinamičku alokaciju prostora za pamćenje matrice istog formata i istog tipa elemenata kao što je i matrica proslijeđena kao parametar, zatim da kopira sadržaj matrice proslijeđene kao parametar u tako alociran prostor, i konačno, da vrati kao rezultat dvojni pokazivač koji omogućava pristup elementima tako kreirane matrice. U slučaju da alokacija ne uspije, funkcija treba da baci tekst "Kreiranje nije uspjelo" kao izuzetak. Također, funkcija se treba pobrinuti da ni u koj slučaju ne može doći do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju, i eventualno uhvatili izuzetak koji bi mogao biti bačen iz nje.

Rješenje:

```
template <typename Tip>
Tip **Alokacija(const vector<vector<Tip> > &v) {
  Tip **mat(0);
  try {
    mat = new Tip*[v.size()];
    for(int i = 0; i < v.size(); i++) mat[i] = 0;
    for(int i = 0; i < v.size(); i++) {
      mat[i] = new Tip[v[i].size()];
      for(int j = 0; j < v[i].size(); j++) mat[i][j] = v[i][j];
    }
  }
  catch(...) {
    for(int i = 0; i < v.size(); i++) delete[] mat[i];
    delete[] mat;
    throw "Alokacija nije uspjela!";
  }
  return mat;
}

...
vector<vector<double> > v;
...
try {
  double **m = Alokacija(v);
  ...
}
catch(const char poruka[]) {
  cout << poruka << endl;
}
```

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" SA RJEŠENJIMA (GRUPA B)

Napomena: Razumije se da sva prikazana rješenja (osim za Zadatak 1) predstavljaju samo jedno od mnoštva mogućih ispravnih rješenja. Prikazano rješenje je tipično najkraće od svih mogućih rješenja, ali ne uvijek nužno i najefikasnije.

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa, uz kratko obrazloženje zbog čega su rezultati onakvi kakvi jesu. Oprez: bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
#include <string>
#include <cctype>
#include <algorithm>
using namespace std;
void F1(int a, int b) {
  a = b - 2; b = a - 2;
  cout << a << b;
}
void F2(int &a, int &b) {
  a = b - 2; b = a - 2;
  cout << a << b;
}
int *F3(int a, int *b) {
  cout << (*b)++;
  return new int(a);
}
int main() {
  int x(9);
  complex<double> c(2), i(0, 1);
  cout << setw(7) << c * c << c * i << setw(10) << c * i - c << endl;
  string s("asdfghjklyxcvbnm");
  F1(x, x);
  cout << x << endl;
  F2(x, x);
  cout << x << endl;
  *F3(x, &x) = 10;
  cout << x << endl;
  transform(&s[8], &s[14], &s[5], (int(*)(int))toupper);
  cout << setw(4) << s.length() + s.size() << " " << s << endl;
  return 0;
}
```

Napomena: funkcija "`transform`" transformira blok omeđen sa prva dva parametra tako što na svaki element bloka primjenjuje funkciju zadanu četvrtim parametrom, dok rezultate funkcije smješta u blok čiji je početak određen trećim parametrom. Funkcija "`toupper`" pretvara mala slova u velika.

Rješenje:

```
□□(4,0)(0,2)□□□□(-2,2)
759
555
56
□□32□asdfgLYXCVBcvbnm
```

Zadatak 2 (2,5 poen)

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor u koji su prepisani samo oni elementi primljenog vektora koji su složeni brojevi (tj. koji nisu prosti). Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 7, 9, 2, 13, 23, 25, 17 i 91 funkcija treba da vrati kao rezultat vektor čiji su elementi 9, 25 i 91.

Napomena: U svim zadacima gdje nije drugačije rečeno, napišite *samo funkciju*, i ništa više.

Rješenje:

```
vector<int> IzdvojiSlozene(const vector<int> &v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    bool slozen(false);
    for(int j = 2; j < sqrt(v[i]); j++)
      if(v[i] % j == 0) slozen = true;
    if(slozen) v1.push_back(v[i]);
  }
  return v1;
}
```

Zadatak 3 (2 poena)

Napišite funkciju "`AnalizaStringa`" sa četiri parametra. Prvi parametar je neki dinamički string (tj. objekat tipa "`string`"). Funkcija treba da utvrdi koliko u tom stringu ima znakova koji su slova (bilo velika bilo mala), zatim znakova koji su cifre, kao i ostalih znakova (tj. znakova koji nisu niti slova niti cifre), i da smjesti rezultate analize redom u drugi, treći i četvrti parametar respektivno. Napišite i kratki isječak programa u kojem ćete demonstrirati kako bi se mogla upotrijebiti napisana funkcija.

Rješenje:

```
void AnalizaStringa(const string &s, int &br_s, int &br_c, int &br_o) {
  br_s = br_c = br_o = 0;
  for(int i = 0; i < s.length(); i++)
    if(s[i] >= 'A' && s[i] <= 'Z' || s[i] >= 'a' && s[i] <= 'z') br_s++;
    else if(s[i] >= '0' && s[i] <= '9') br_c++;
    else br_o++;
}

...
int br_slova, br_cifri, br_ostalih;
AnalizaStringa("sdJHJKsrhUgsZ29rej!!?", br_slova, br_cifri, br_ostalih);
cout << br_slova << " " << br_cifri << " " << br_ostalih;
```

Zadatak 4 (1,5 poen)

Napišite funkciju sa jednim cjelobrojnim parametrom koja kao rezultat vraća najveću cifru tog broja. Na primjer, ukoliko se funkciji proslijedi broj 35842, funkcija treba da kao rezultat vrati broj 8.

Rješenje:

```
int NajvecaCifra(int broj) {
  int najveca(0);
  while(broj > 0) {
    if(broj % 10 > najveca) najveca = broj % 10;
    broj /= 10;
  }
  return najmanja;
}
```

Zadatak 5 (2 poena)

Napišite generičku funkciju sa 3 parametra *x*, *f* i *n*. *x* je vrijednost nekog nedefiniranog tipa T, *f* je funkcija koja prima parametar tipa T i vraća rezultat tipa T, dok je *n* cijeli broj. Funkcija treba da kao rezultat vrati vektor sa *n* elemenata čiji su elementi redom $f(x)$, $f(f(x))$, $f(f(f(x)))$, ... pa sve do $f(f(f(...(f(x))...))$ gdje se u posljednjem slučaju funkcija *f* uzastopno primjenjuje *n* puta.

Rješenje:

```
template <typename T>
vector<T> Kaskada(T x, T f(T), int n) {
  vector<T> v;
  for(int i = 1; i <= n; i++) v.push_back(x = f(x));
  return v;
}
```

Zadatak 6 (2 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`equal`". Ova funkcija vraća kao rezultat logičku logičku vrijednost "**`true`**" ukoliko je blok elemenata između pokazivača *p1* i *p2* identičan po sadržaju bloku elemenata na koji pokazuje pokazivač *p3*, a u suprotnom vraća logičku vrijednost "**`false`**". Napišite sami generičku funkciju "`RazlicitiBlokovi`" koja radi suprotnu stvar u odnosu na funkciju "`equal`", odnosno vraća "**`true`**" ukoliko se blokovi razlikuju, a "**`false`**" ukoliko su identički (pri tome, za realizaciju *ne smijete* koristiti funkciju "`equal`").

Rješenje:

```
template <typename Pokazivac>
bool RazlicitiBlokovi(Pokazivac p1, Pokazivac p2, Pokazivac p3) {
  while(p1 != p2)
    if(*p1++ != *p3++) return true;
  return false;
}
```

Zadatak 7 (2 poena)

Pretpostavimo da želimo sortirati niz kompleksnih brojeva tako da kompleksni broj z_1 dolazi ispred kompleksnog broja z_2 ako i samo ako je realni dio od z_1 veći od realnog dijela od z_2, ili ako su realni dijelovi od z_1 i z_2 jednaki, ali je imaginarni dio od z_1 veći od imaginarnog dijela od z_2. Definirajte odgovarajuću funkciju kriterija, i napišite isječak programa u kojem ćete pokazati kako biste deklarirali niz kompleksnih brojeva, napunili ga vrijednostima unesenim sa tastature i sortirali ga u traženi poredak korištenjem funkcije "`sort`" iz biblioteke "`algorithm`".

Napomena: Funkcija "`sort`" prima kao parametre dva pokazivača koji omeđuju blok, i opcionalno, funkciju kriterija kao treći parametar. Realni i imaginarni dio kompleksnog broja mogu se dobiti pozivom funkcija "`real`" i "`imag`" respektivno (prosljeđujući im kompleksni broj kao parametar).

Rješenje:

```
bool Kriterij(complex<double> z1, complex<double> z2) {
  return real(z1) > real(z2) ||
    real(z1) == real(z2) && imag(z1) < imag(z2);
}

...
complex<double> niz[100];
for(int i = 0; i <= 100; i++) cin >> niz[i];
sort(niz, niz + 100, Kriterij);
```

Zadatak 8 (3 poena)

Napišite generičku funkciju koja prima kao parametar matricu organiziranu kao vektor vektora, čiji su elementi proizvoljnog tipa. Funkcija treba da izvrši dinamičku alokaciju prostora za pamćenje matrice istog formata i istog tipa elemenata kao što je i matrica proslijeđena kao parametar, zatim da kopira sadržaj matrice proslijeđene kao parametar u tako alociran prostor, i konačno, da vrati kao rezultat dvojni pokazivač koji omogućava pristup elementima tako kreirane matrice. U slučaju da alokacija ne uspije, funkcija treba da baci tekst "Neuspješna alokacija" kao izuzetak. Također, funkcija se treba pobrinuti da ni u koj slučaju ne može doći do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju, i eventualno uhvatili izuzetak koji bi mogao biti bačen iz nje.

Rješenje:

```
template <typename Tip>
Tip **Alokacija(const vector<vector<Tip> > &v) {
  Tip **mat(0);
  try {
    mat = new Tip*[v.size()];
    for(int i = 0; i < v.size(); i++) mat[i] = 0;
    for(int i = 0; i < v.size(); i++) {
      mat[i] = new Tip[v[i].size()];
      for(int j = 0; j < v[i].size(); j++) mat[i][j] = v[i][j];
    }
  }
  catch(...) {
    for(int i = 0; i < v.size(); i++) delete[] mat[i];
    delete[] mat;
    throw "Neuspješna alokacija!";
  }
  return mat;
}

...
vector<vector<double> > v;
...
try {
  double **m = Alokacija(v);
  ...
}
catch(const char poruka[]) {
  cout << poruka << endl;
}
```

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA A)

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <algorithm>

using namespace std;

void F1(int a, int b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

void F2(int &a, int &b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

int &F3(int *a, int *b) {
  (*a)++;
  return *b;
}

int main() {
  int a(2), b(5);
  string s("qwertzuiop");
  F1(b, a);
  cout << a << b << endl;
  F2(b, a);
  cout << a << b << endl;
  F3(&b, &a) = 10;
  cout << a << b << endl;
  reverse(&s[2], &s[6]);
  cout << setw(3) << s.length() << setw(12) << s << s.size() << endl;
  return 0;
}
```

Rješenje:

```
523-125
523-1-13
104
□10□□qwztreuiop10
```

Zadatak 2 (2 poena)

Data je generička funkcija

```
template <typename IspisiviTip>
void F(IspisiviTip t) {
  cout << t;
}
```

Odredite kojeg će tačno tipa biti metatip "`IspisiviTip`" u svakom od dolje navedenih slučajeva ukoliko se izvrše sljedeći pozivi:

a) `F(3);` b) `F<float>(3);` c) `F(3.);` d) `F('3');`
e) `F<int>('3');` f) `F("3");` g) `F("333");` h) `F<string>("3");`

Rješenje:

a) **`int`** b) **`float`** c) **`double`** d) **`char`**
e) **`int`** f) **`char[]`** (ili **`char*`**) g) **`char[]`** (ili **`char*`**) h) `string`

Napomena: Umjesto "**`char[]`**" ili "**`char*`**", još precizniji odgovor bio bi "**`const char[]`**" ili "**`const char*`**", ali na tome se nije insistiralo.

Zadatak 3 (2 poena)

Objasnite ukratko u čemu je razlika između *djelimične* (*parcijalne*) i *potpune dedukcije tipa* kod generičkih funkcija. Razliku ilustrirajte na nekom jednostavnom primjeru.

Rješenje:

Kod *djelimične* (*parcijalne*) *dedukcije tipa,* neke informacije o tipu su *poznate*, na primjer da je tip neki niz, ili vektor, ali se ne zna šta su mu elementi, ili neki pokazivač, ali se ne zna koji je tip objekta na koji pokazivač pokazuje. Na primjer, sljedeće tri funkcije koriste djelimičnu dedukciju tipa:

```
template <typename Tip>
void F1(Tip niz[]) {
  cout << niz[0];
}
```

```
template <typename Tip>
void F2(vector<Tip> v) {
  cout << v[0];
}
```

```
template <typename Tip>
void F3(Tip *p) {
  cout << *p;
}
```

Kod *potpune dedukcije tipa*, nikakve apriorne informacije o tipu se ne pretpostavljaju, tako da se funkcije mogu primjenjivati na širu skupinu argumenata, ali je potreban veći oprez. Na primjer, sljedeće dvije funkcije koriste potpunu dedukciju tipa. Pri tome se funkcija F4 može koristiti i sa nizovima i sa vektorima (i sa dekovima, i bilo čime što se ponaša slično njima), dok se funkcija F5 može koristiti i sa pokazivačima i sa iteratorima (i sa bilo čime što se ponaša slično njima):

```
template <typename KontejnerskiTip>
void F1(KontejnerskiTip t) {
  cout << t[0];
}
```

```
template <typename PokazivackiTip>
void F5(PokazivackiTip p) {
  cout << *p;
}
```

Zadatak 4 (2 poena)

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi sume cifara odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 32, 459, 72, 8, 24, 771 i 13, funkcija treba da kao rezultat vrati vektor čiji su elementi 5, 18, 9, 8, 6, 15 i 4.

Rješenje (jedno od mogućih):

```
vector<int> SumeCifara(const vector<int> &v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    int n(v[i]), suma(0);
    while(n > 0) {
      suma += n % 10; n /= 10;
    }
    v1.push_back(suma);
  }
```

```
  return v1;
}
```

Napomena: Mnogo je (ispravnih) rješenja bilo poput sljedećeg:

```
vector<int> SumeCifara(vector<int> v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    int suma(0);
    while(v[i] > 0) {
      suma += v[i] % 10; v[i] /= 10;
    }
    v1.push_back(suma);
  }
  return v1;
}
```

Mada je ovo rješenje ispravno, ono *nije ispravno*, ukoliko se formalni parametar "v" deklarira kao referenca (bilo obična, bilo referenca na konstantni objekat). Naime, ovako izvedena funkcija *mijenja* sadržaj elemenata vektora "v". Stoga, ukoliko se "v" deklarira kao referenca (obična), poziv ove funkcije ostaviće trag na vektoru koji se zada kao stvarni parametar pri pozivu funkcije, a ukoliko se "v" deklarira kao referenca na konstantni objekat, kompajler će prijaviti grešku, jer promjena vektora "v" neće biti dozvoljena. Stoga, ovo rješenje je ispravno samo ako "v" *nije referenca*. Međutim, prethodno rješenje je u svakom slučaju efikasnije, jer ne dolazi do nepotrebnog kopiranja vektora.

Zadatak 5 (2 poena)

Napišite funkciju koja kao parametar prima neki string, a koja transformira taj string u novi string kod kojeg su ASCII šifre svih znakova uvećane za 1 u odnosu na izvorni string (sama funkcija ne vraća *nikakav rezultat*). Na primjer, ukoliko se funkciji proslijedi string "s" čiji je sadržaj "`proba`", sadržaj stringa nakon poziva funkcije treba da bude "`qspcb`".

Rješenje (jedno od mogućih):

```
void Transformiraj(string &s) {
  for(int i = 0; i < s.length(); i++) s[i]++;
}
```

Napomena: Formalni parametar "s" *mora biti referenca*. Dalje, eksplicitne pretvorbe tipa poput "`s[i] = char(int(s[i]) + 1)`" nisu neispravne, ali *nisu ni potrebne*.

Zadatak 6 (2 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`find_if`". Ova funkcija vraća kao rezultat pokazivač na prvi element u bloku između pokazivača *p1* i *p2* za koje funkcija *f* vraća kao rezultat "**`true`**" kad joj se proslijedi kao argument (ili *p*2 ukoliko takav element ne postoji), pri čemu je sintaksa poziva ove funkcije "`find_if(`*p1*`, `*p2*`, `*f*`)`". Napišite sami generičku funkciju "`Nadji`" koja prima potpuno iste parametre i obavlja istu funkciju kao i funkcija "`find_if`".

Rješenje (jedno od mogućih):

```
template <typename NekiTip>
NekiTip *Nadji(NekiTip *p1, NekiTip *p2, bool f(NekiTip)) {
  for(; p1 < p2; p1++)
    if(f(*p1)) return p1;
  return p2;
}
```

Napomena: Ovako napisana funkcija koristi *djelimičnu dedukciju tipa*, i stoga se može primijeniti *samo na pokazivače*, a ne i na iteratore. Prava funkcija "`find_if`" radi i sa iteratorima, što bi se moglo postići *potpunom dedukcijom tipa*, kao u sljedećoj izvedbi (ovdje "`PokazivackiTip`" predstavlja neki tip koji se ponaša *poput pokazivača*, što uključuje i iteratore):

```
template <typename NekiTip, typename PokazivackiTip>
PokazivackiTip Nadji(PokazivackiTip p1, PokazivackiTip p2, bool f(NekiTip)) {
  for(; p1 < p2; p1++)
    if(f(*p1)) return p1;
  return p2;
}
```

Zadatak 7 (2 poena)

Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz realnih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj pozitivnih i broj negativnih elemenata u nizu. Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost "**`true`**" ako i samo ako je makar jedan element niza jednak nuli (inače vraća logičku vrijednost "**`false`**"). Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati samo kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

Rješenje (jedno od mogućih):

```
bool PozNeg(double niz[], int br_el, int &br_poz, int &br_neg) {
  bool ima_nula(false);
  br_poz = br_neg = 0;
  for(int i = 0; i < br_el; i++) {
    if(niz[i] > 0) br_poz++;
    else if(niz[i] > 0) br_neg++;
    else ima_nula = true;
  }
  return ima_nula;
}
```

Isječak glavnog programa:

```
double a[5] = {-2.3, 4, 5.2, 0, 8.6};
int brp, brn;
bool ima = PozNeg(a, 5, brp, brn);
if(ima) cout << "Ima nula\n";
else cout << "Nema nula\n";
cout << "Pozitivnih: " << brp << " Negativnih: " << brn << endl;
```

Napomena: Formalni parametri "`br_poz`" i "`br_neg`" *moraju biti reference*.

Zadatak 8 (3 poena)

Napišite generičku funkciju koja vrši dinamičku alokaciju prostora za pamćenje kvadratne matrice formata $n \times n$, pri čemu je n jedan od parametara funkcije (elementi matrice mogu biti proizvoljnog tipa). Parametri funkcije su referenca na dvojni pokazivač koji će služiti za pristup tako kreiranom prostoru, kao i dimenzija matrice n. Funkcija ne vraća nikakav rezultat, nego samo pridružuje adresu alociranog prostora pokazivaču koji joj je proslijeđen kao parametar. U slučaju da alokacija ne uspije, funkcija treba da baci tekst "Kreiranje nije uspjelo" kao izuzetak. Također, funkcija se treba pobrinuti da ni u koj slučaju ne može doći do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju, i eventualno uhvatiti izuzetak koji bi mogao biti bačen iz nje.

Rješenje (jedno od mogućih):

```
template <typename TipElemenata>
void KreirajMatricu(TipElemenata **&mat, int n) {
  mat = 0;
  try {
    mat = new TipElemenata*[n];
    for(int i = 0; i < n; i++) mat[i] = 0;
    for(int i = 0; i < n; i++) mat[i] = new TipElemenata[n];
  }
  catch(...) {
    for(int i = 0; i < n; i++) delete[] mat[i];
    delete[] mat;
    throw "Kreiranje nije uspjelo!";
  }
}
```

Isječak glavnog programa:

```
double **m(0);
try {
  KreirajMatricu(m, 10)
  ...
}
catch(const char poruka[]) {
  cout << poruka << endl;
}
```

Napomena: Ponuđeno rješenje je vjerovatno najjednostavnije, jer koristi samo jedan "**try**" – "**catch**" blok u funkciji. Moguće je napisati dosta ispravnih rješenja, koji sadrže dva "**try**" – "**catch**" bloka, pri čemu se u spoljašnjem bloku hvata izuzetak bačen eventualno pri alokaciji niza pokazivača, a u unutrašnjem bloku se hvataju izuzeci eventualno bačeni pri alokaciji individualnih redova matrice. Uglavnom, *tipične greške* koje se mogu napraviti su: nehvatanje nekog od izuzetaka koji bi mogli biti bačeni (najčešće onog koji nastaje pri alokaciji niza pokazivača), zatim neoslobađanje zauzete memorije u slučaju djelomično uspjele alokacije, kao i primjena "**delete[]**" operatora na pokazivače koji nemaju definiranu vrijednost.

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA B)

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <algorithm>

using namespace std;

void F1(int a, int b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

void F2(int &a, int &b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

int &F3(int *a, int *b) {
  (*a)++;
  return *b;
}

int main() {
  int a(4), b(3);
  string s("asdfghjkl");
  F1(b, a);
  cout << a << b << endl;
  F2(b, a);
  cout << a << b << endl;
  F3(&b, &a) = 10;
  cout << a << b << endl;
  reverse(&s[4], &s[8]);
  cout << setw(2) << s.length() << setw(13) << s << s.size() << endl;
  return 0;
}
```

Rješenje:

```
34-1543
34-155-1
100
□9□□□□asdfkjhgl9
```

Zadatak 2 (2 poena)

Data je generička funkcija

```
template <typename IspisiviTip>
void F(IspisiviTip t) {
  cout << t;
}
```

Odredite kojeg će tačno tipa biti metatip "`IspisiviTip`" u svakom od dolje navedenih slučajeva ukoliko se izvrše sljedeći pozivi:

a) `F(5.);` b) `F<float>(5);` c) `F(55);` d) `F(int('5'));`
e) `F('5');` f) `F<string>("5");` g) `F("5");` g) `F("555");`

Rješenje:

a) `double` b) `float` c) `int` d) `int`
e) `char` f) `string` g) `char[]` (ili `char*`) h) `char[]` (ili `char*`)

Napomena: Umjesto "`char[]`" ili "`char*`", još precizniji odgovor bio bi "`const char[]`" ili "`const char*`", ali na tome se nije insistiralo.

Zadatak 3 (2 poena)

Objasnite ukratko u kojim slučajevima se ne može koristiti *mehanizam dedukcije tipa* kod generičkih funkcija, nego se mora koristiti *eksplicitna specifikacija tipa*. Ilustrirajte ovo na nekom jednostavnom primjeru.

Rješenje:

Eksplicitna specifikacija tipa se mora koristiti u slučajevima kada kompajler nema načina da na osnovu poziva funkcije odredi kojim stvarnim tipovima odgovaraju metatipovi u deklaraciji generičke funkcije. To se može desiti u dva slučaja. Prvi slučaj je kada se neki metatip uopće ne javlja u specifikaciji formalnih parametara funkcije, nego se koristi kao tip povratne vrijednosti, ili negdje unutar tijela funkcije (recimo, za deklaraciju neke lokalne promjenljive). Na primjer, pri pozivu sljedeće dvije funkcije mora se koristiti eksplicitna specifikacija tipa:

```
template <typename Tip>
Tip F1() {
   ...
}
```

```
template <typename Tip>
void F2() {
  Tip t;
  ...
}
```

Drugi slučaj nastaje ukoliko funkcija ima više formalnih parametara koji su istog metatipa, a pri pozivu funkcije navedemo stvarne parametre različitih tipova. U tom slučaju kompajler ne može odrediti koji od više mogućih tipova treba pridružiti metatipu, pa se opet mora koristiti eksplicitna specifikacija. Na primjer, ukoliko sljedeću funkciju pozovemo navodeći dva stvarna parametra različitih tipova, eksplicitna specifikacija biće neophodna:

```
template <typename Tip>
void F3(Tip t1, Tip t2) {
   ...
}
```

Zadatak 4 (2 poena)

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi najmanje cifre odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 32, 459, 72, 8, 24, 771 i 13, funkcija treba da kao rezultat vrati vektor čiji su elementi 2, 4, 2, 8, 2, 1 i 1.

Rješenje (jedno od mogućih):

```
vector<int> NajmanjeCifre(const vector<int> &v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    int n(v[i]), min_c(9);
```

```
    while(n > 0) {
      if(n % 10 < min_c) min_c = n % 10;
      n /= 10;
    }
    v1.push_back(min_c);
  }
  return v1;
}
```

Napomena: Moguća su i brojna druga ispravna rješenja. Međutim, vrijedi analogna napomena kao u rješenju Zadatka 4. u Grupi A.

Zadatak 5 (2 poena)

Napišite funkciju koja kao parametar prima neki string, a koja transformira taj string u novi string kod kojeg su ASCII šifre svih znakova umanjene za 1 u odnosu na izvorni string (sama funkcija ne vraća *nikakav rezultat*). Na primjer, ukoliko se funkciji proslijedi string "`s`" čiji je sadržaj "`proba`", sadržaj stringa nakon poziva funkcije treba da bude "`oqna@`".

Rješenje (jedno od mogućih):

```
void Transformiraj(string &s) {
  for(int i = 0; i < s.length(); i++) s[i]--;
}
```

Napomena: Formalni parametar "`s`" *mora biti referenca*. Dalje, eksplicitne pretvorbe tipa poput "`s[i] = char(int(s[i]) - 1)`" nisu neispravne, ali *nisu ni potrebne*.

Zadatak 6 (2 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`count_if`". Ova funkcija vraća kao rezultat broj elemenata u bloku između pokazivača *p1* i *p2* za koje funkcija *f* vraća kao rezultat "**`true`**" kad joj se proslijede kao argument, pri čemu je sintaksa poziva ove funkcije "`count_if(`*p1*`, `*p2*`, `*f*`)`". Napišite sami generičku funkciju "`Prebroji`" koja prima potpuno iste parametre i obavlja istu funkciju kao i funkcija "`count_if`".

Rješenje (jedno od mogućih):

```
template <typename NekiTip>
int Prebroji(NekiTip *p1, NekiTip *p2, bool f(NekiTip)) {
  int brojac(0);
  for(; p1 < p2; p1++)
    if(f(*p1)) brojac++;
  return brojac;
}
```

Napomena: Ovako napisana funkcija koristi *djelimičnu dedukciju tipa*, i stoga se može primijeniti *samo na pokazivače*, a ne i na iteratore. Prava funkcija "`count_if`" radi i sa iteratorima, što bi se moglo postići *potpunom dedukcijom tipa*, kao u sljedećoj izvedbi (ovdje "`PokazivackiTip`" predstavlja neki tip koji se ponaša *poput pokazivača*, što uključuje i iteratore):

```
template <typename NekiTip, typename PokazivackiTip>
int Prebroji(PokazivackiTip p1, PokazivackiTip p2, bool f(NekiTip)) {
  int brojac(0);
  for(; p1 < p2; p1++)
    if(f(*p1)) brojac++;
  return brojac;
}
```

Zadatak 7 (2 poena)

Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz cijelih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj parnih i broj neparnih elemenata u nizu. Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost "**`true`**" ako i samo ako je makar jedan element niza negativan (inače vraća logičku vrijednost "**`false`**"). Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati samo kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

Rješenje (jedno od mogućih):

```
bool ParNepar(int niz[], int br_el, int &br_par, int &br_nepar) {
  bool ima_neg(false);
  br_par = br_nepar = 0;
  for(int i = 0; i < br_el; i++) {
    if(niz[i] % 2 == 0) br_par++;
    else br_nepar++;
    if(niz[i] < 0) ima_neg = true;
  }
  return ima_neg;
}
```

Isječak glavnog programa:

```
int a[5] = {3, 4, -1, 2, 6};
int brp, brn;
bool ima = ParNepar(a, 5, brp, brn);
if(ima) cout << "Ima negativnih\n";
else cout << "Nema negativnih\n";
cout << "Parnih: " << brp << " Neparnih: " << brn << endl;
```

Napomena: Formalni parametri "`br_par`" i "`br_nepar`" *moraju biti reference.*

Zadatak 8 (3 poena)

Napišite generičku funkciju koja vrši dinamičku alokaciju prostora za pamćenje kvadratne matrice formata $n \times n$, pri čemu je n jedan od parametara funkcije (elementi matrice mogu biti proizvoljnog tipa). Parametri funkcije su referenca na dvojni pokazivač koji će služiti za pristup tako kreiranom prostoru, kao i dimenzija matrice n. Funkcija ne vraća nikakav rezultat, nego samo pridružuje adresu alociranog prostora pokazivaču koji joj je prosljeđen kao parametar. U slučaju da alokacija ne uspije, funkcija treba da baci tekst "Alokacija nije obavljena" kao izuzetak. Također, funkcija se treba pobrinuti da ni u koj slučaju ne može doći do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju, i eventualno uhvatiti izuzetak koji bi mogao biti bačen iz nje.

Rješenje (jedno od mogućih):

```
template <typename TipElemenata>
void KreirajMatricu(TipElemenata **&mat, int n) {
  mat = 0;
  try {
    mat = new TipElemenata*[n];
    for(int i = 0; i < n; i++) mat[i] = 0;
    for(int i = 0; i < n; i++) mat[i] = new TipElemenata[n];
  }
  catch(...) {
    for(int i = 0; i < n; i++) delete[] mat[i];
    delete[] mat;
```

```
    throw "Alokacija nije obavljena!";
  }
}
```

Isječak glavnog programa:

```
double **m(0);
try {
  KreirajMatricu(m, 10)
  ...
}
catch(const char poruka[]) {
  cout << poruka << endl;
}
```

Napomena: Vrijede iste napomene kao u rješenju Zadatka 8. u Grupi A.

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA C)

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <algorithm>

using namespace std;

void F1(int a, int b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

void F2(int &a, int &b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

int &F3(int *a, int *b) {
  (*a)++;
  return *b;
}

int main() {
  int a(5), b(1);
  string s("yxcvbnm");
  F1(b, a);
  cout << a << b << endl;
  F2(b, a);
  cout << a << b << endl;
  F3(&b, &a) = 8;
  cout << a << b << endl;
  reverse(&s[1], &s[5]);
  cout << setw(4) << s.length() << setw(10) << s << s.size() << endl;
  return 0;
}
```

Rješenje:

```
15-4951
15-499-4
8-3
□□□7□□□ybvcxnm7
```

Zadatak 2 (2 poena)

Data je generička funkcija

```
template <typename IspisiviTip>
void F(IspisiviTip t) {
  cout << t;
}
```

Odredite kojeg će tačno tipa biti metatip "`IspisiviTip`" u svakom od dolje navedenih slučajeva ukoliko se izvrše sljedeći pozivi:

a) `F<float>(2);` b) `F(3.);` c) `F(2);` d) `F('2');`
e) `F("2");` f) `F("22");` g) `F<int>('2');` h) `F<string>("2");`

Rješenje:

a) `float` b) `double` c) `int` d) `char`
e) `char[]` (ili `char*`) f) `char[]` (ili `char*`) g) `int` h) `string`

Napomena: Umjesto "`char[]`" ili "`char*`", još precizniji odgovor bio bi "`const char[]`" ili "`const char*`", ali na tome se nije insistiralo.

Zadatak 3 (2 poena)

Objasnite ukratko u kojim slučajevima se ne može koristiti *mehanizam dedukcije tipa* kod generičkih funkcija, nego se mora koristiti *eksplicitna specifikacija tipa*. Ilustrirajte ovo na nekom jednostavnom primjeru.

Rješenje:

Eksplicitna specifikacija tipa se mora koristiti u slučajevima kada kompajler nema načina da na osnovu poziva funkcije odredi kojim stvarnim tipovima odgovaraju metatipovi u deklaraciji generičke funkcije. To se može desiti u dva slučaja. Prvi slučaj je kada se neki metatip uopće ne javlja u specifikaciji formalnih parametara funkcije, nego se koristi kao tip povratne vrijednosti, ili negdje unutar tijela funkcije (recimo, za deklaraciju neke lokalne promjenljive). Na primjer, pri pozivu sljedeće dvije funkcije mora se koristiti eksplicitna specifikacija tipa:

```
template <typename Tip>
Tip F1() {
   ...
}
```

```
template <typename Tip>
void F2() {
  Tip t;
  ...
}
```

Drugi slučaj nastaje ukoliko funkcija ima više formalnih parametara koji su istog metatipa, a pri pozivu funkcije navedemo stvarne parametre različitih tipova. U tom slučaju kompajler ne može odrediti koji od više mogućih tipova treba pridružiti metatipu, pa se opet mora koristiti eksplicitna specifikacija. Na primjer, ukoliko sljedeću funkciju pozovemo navodeći dva stvarna parametra različitih tipova, eksplicitna specifikacija biće neophodna:

```
template <typename Tip>
void F3(Tip t1, Tip t2) {
   ...
}
```

Zadatak 4 (2 poena)

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi produkti cifara odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 32, 459, 72, 8, 24, 771 i 13, funkcija treba da kao rezultat vrati vektor čiji su elementi 6, 180, 14, 8, 8, 49 i 3.

Rješenje (jedno od mogućih):

```
vector<int> ProduktiCifara(const vector<int> &v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    int n(v[i]), produkt(1);
```

```
    while(n > 0) {
      produkt *= n % 10; n /= 10;
    }
    v1.push_back(produkt);
  }
  return v1;
}
```

Napomena: Vrijedi analogna napomena kao u rješenju Zadatka 4. u Grupi A.

Zadatak 5 (2 poena)

Napišite funkciju koja kao parametar prima neki string, a koja transformira taj string u novi string kod kojeg su ASCII šifre svih znakova uvećane za 2 u odnosu na izvorni string (sama funkcija ne vraća *nikakav rezultat*). Na primjer, ukoliko se funkciji proslijedi string "`s`" čiji je sadržaj "`proba`", sadržaj stringa nakon poziva funkcije treba da bude "`rtqdc`".

Rješenje (jedno od mogućih):

```
void Transformiraj(string &s) {
  for(int i = 0; i < s.length(); i++) s[i] += 2;
}
```

Napomena: Formalni parametar "`s`" *mora biti referenca*. Dalje, eksplicitne pretvorbe tipa poput "`s[i] = char(int(s[i]) + 2)`" nisu neispravne, ali *nisu ni potrebne*.

Zadatak 6 (2 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`replace_if`". Ova funkcija zamjenjuje sve elemente između pokazivača *p1* i *p2* za koje funkcija *f* vraća kao rezultat "**`true`**" kad joj se proslijede kao argument, sa elementima sa vrijednošću *v*, pri čemu je sintaksa poziva ove funkcije "`replace_if(`*p1*`,` *p2*`,` *f*`,` *v*`)`". Napišite sami generičku funkciju "`Zamijeni`" koja prima potpuno iste parametre i obavlja istu funkciju kao i funkcija "`replace_if`".

Rješenje (jedno od mogućih):

```
template <typename NekiTip>
void Zamijeni(NekiTip *p1, NekiTip *p2, bool f(NekiTip), NekiTip v) {
  for(; p1 < p2; p1++)
    if(f(*p1)) *p1 = v;
}
```

Napomena: Ovako napisana funkcija koristi *djelimičnu dedukciju tipa*, i stoga se može primijeniti *samo na pokazivače*, a ne i na iteratore. Prava funkcija "`replace_if`" radi i sa iteratorima, što bi se moglo postići *potpunom dedukcijom tipa*, kao u sljedećoj izvedbi (ovdje "`PokazivackiTip`" predstavlja neki tip koji se ponaša *poput pokazivača*, što uključuje i iteratore):

```
template <typename NekiTip, typename PokazivackiTip>
void Zamijeni(PokazivackiTip p1, PokazivackiTip p2, bool f(NekiTip),
  NekiTip v) {
    for(; p1 < p2; p1++)
      if(f(*p1)) *p1 = v;
}
```

Zadatak 7 (2 poena)

Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz realnih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj cjelobrojnih elemenata i broj elemenata koji nisu cijeli u nizu. Pored toga, funkcija treba da vrati kao

rezultat logičku vrijednost "**true**" ako i samo ako je makar jedan element niza negativan broj (inače vraća logičku vrijednost "**false**"). Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati samo kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

Rješenje (jedno od mogućih):

```
bool Cjelobrojni(double niz[], int br_el, int &br_c, int &br_nc) {
  bool ima_neg(false);
  br_c = br_nc = 0;
  for(int i = 0; i < br_el; i++) {
    if(niz[i] == int(niz[i])) br_c++;
    else br_nc++;
    if(niz[i] < 0) ima_neg = true;
  }
  return ima_neg;
}
```

Isječak glavnog programa:

```
double a[5] = {3.5, 4, -1.2, 2.7, 5};
int brc, brnc;
bool ima = Cjelobrojni(a, 5, brc, brnc);
if(ima) cout << "Ima negativnih\n";
else cout << "Nema negativnih\n";
cout << "Cijelih: " << brc << " Necijelih: " << brnc << endl;
```

Napomena: Formalni parametri "`br_c`" i "`br_nc`" *moraju biti reference*.

Zadatak 8 (3 poena)

Napišite generičku funkciju koja vrši dinamičku alokaciju prostora za pamćenje kvadratne matrice formata $n \times n$, pri čemu je n jedan od parametara funkcije (elementi matrice mogu biti proizvoljnog tipa). Parametri funkcije su referenca na dvojni pokazivač koji će služiti za pristup tako kreiranom prostoru, kao i dimenzija matrice n. Funkcija ne vraća nikakav rezultat, nego samo pridružuje adresu alociranog prostora pokazivačk koji joj je prosljeđen kao parametar. U slučaju da alokacija ne uspije, funkcija treba da baci tekst "Nema dovoljno memorije" kao izuzetak. Također, funkcija se treba pobrinuti da ni u koj slučaju ne može doći do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju, i eventualno uhvatiti izuzetak koji bi mogao biti bačen iz nje.

Rješenje (jedno od mogućih):

```
template <typename TipElemenata>
void KreirajMatricu(TipElemenata **&mat, int n) {
  mat = 0;
  try {
    mat = new TipElemenata*[n];
    for(int i = 0; i < n; i++) mat[i] = 0;
    for(int i = 0; i < n; i++) mat[i] = new TipElemenata[n];
  }
  catch(...) {
    for(int i = 0; i < n; i++) delete[] mat[i];
    delete[] mat;
    throw "Nema dovoljno memorije!";
  }
}
```

Isječak glavnog programa:

```
double **m(0);
try {
  KreirajMatricu(m, 10)
  ...
}
catch(const char poruka[]) {
  cout << poruka << endl;
}
```

Napomena: Vrijede iste napomene kao u rješenju Zadatka 8. u Grupi A.

I PARCIJALNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (GRUPA D)

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa. Bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <string>
#include <algorithm>

using namespace std;

void F1(int a, int b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

void F2(int &a, int &b) {
  cout << a << b;
  a -= b; b -= a;
  cout << a << b;
}

int &F3(int *a, int *b) {
  (*a)++;
  return *b;
}

int main() {
  int a(6), b(2);
  string s("qaywsxedc");
  F1(b, a);
  cout << a << b << endl;
  F2(b, a);
  cout << a << b << endl;
  F3(&b, &a) = 10;
  cout << a << b << endl;
  reverse(&s[2], &s[6]);
  cout << setw(4) << s.length() << setw(11) << s << s.size() << endl;
  return 0;
}
```

Rješenje:

```
26-41062
26-41010-4
10-3
□□□9□□qaxswyedc9
```

Zadatak 2 (2 poena)

Data je generička funkcija

```
template <typename IspisiviTip>
void F(IspisiviTip t) {
  cout << t;
}
```

Odredite kojeg će tačno tipa biti metatip "`IspisiviTip`" u svakom od dolje navedenih slučajeva ukoliko se izvrše sljedeći pozivi:

a) `F(0.);` b) `F<float>(0.);` c) `F('0');` d) `F("0");`
e) `F<double>('0');` f) `F("000"));` g) `F<string>("0");` h) `F(0);`

Rješenje:

a) **`double`** b) **`float`** c) **`char`** d) **`char[]`** (ili **`char*`**)
e) **`double`** f) **`char[]`** (ili **`char*`**) g) `string` h) **`int`**

Napomena: Umjesto "**`char[]`**" ili "**`char*`**", još precizniji odgovor bio bi "**`const char[]`**" ili "**`const char*`**", ali na tome se nije insistiralo.

Zadatak 3 (2 poena)

Objasnite ukratko u čemu je razlika između *djelimične* (*parcijalne*) i *potpune dedukcije tipa* kod generičkih funkcija. Razliku ilustrirajte na nekom jednostavnom primjeru.

Rješenje:

Kod *djelimične* (*parcijalne*) *dedukcije tipa,* neke informacije o tipu su *poznate*, na primjer da je tip neki niz, ili vektor, ali se ne zna šta su mu elementi, ili neki pokazivač, ali se ne zna koji je tip objekta na koji pokazivač pokazuje. Na primjer, sljedeće tri funkcije koriste djelimičnu dedukciju tipa:

```
template <typename Tip>
void F1(Tip niz[]) {
  cout << niz[0];
}
```

```
template <typename Tip>
void F2(vector<Tip> v) {
  cout << v[0];
}
```

```
template <typename Tip>
void F3(Tip *p) {
  cout << *p;
 }
```

Kod *potpune dedukcije tipa*, nikakve apriorne informacije o tipu se ne pretpostavljaju, tako da se funkcije mogu primjenjivati na širu skupinu argumenata, ali je potreban veći oprez. Na primjer, sljedeće dvije funkcije koriste potpunu dedukciju tipa. Pri tome se funkcija F4 može koristiti i sa nizovima i sa vektorima (i sa dekovima, i bilo čime što se ponaša slično njima), dok se funkcija F5 može koristiti i sa pokazivačima i sa iteratorima (i sa bilo čime što se ponaša slično njima):

```
template <typename KontejnerskiTip>
void F1(KontejnerskiTip t) {
  cout << t[0];
}
```

```
template <typename PokazivackiTip>
void F5(PokazivackiTip p) {
  cout << *p;
}
```

Zadatak 4 (2 poena)

Napišite funkciju koja kao parametar prima vektor cijelih brojeva, a koja kao rezultat vraća novi vektor čiji su elementi broj cifara odgovarajućih elemenata vektora koji je zadan kao parametar. Na primjer, ukoliko se funkciji proslijedi vektor čiji su elementi 32, 459, 72, 8, 24, 771 i 13, funkcija treba da kao rezultat vrati vektor čiji su elementi 2, 3, 2, 1, 2, 3 i 2.

Rješenje (jedno od mogućih):

```
vector<int> BrojeviCifara(const vector<int> &v) {
  vector<int> v1;
  for(int i = 0; i < v.size(); i++) {
    int n(v[i]), brojac(0);
    if(n == 0) brojac = 1;
    while(n > 0) {
      n /= 10; brojac++;
    }
```

```
    v1.push_back(brojac);
  }
  return v1;
}
```

Napomena: Uvjet "`if(n == 0)`" ubačen je da bi se nula tretirala također kao jednocifren broj (inače će se nula smatrati kao da je broj sa 0 cifara). Ipak, studenti koji nisu uočili ovu anomaliju *nisu gubili poene*. Naravno, moguća su i brojna druga ispravna rješenja. Pored toga, vrijedi analogna napomena kao u rješenju Zadatka 4. u Grupi A.

Zadatak 5 (2 poena)

Napišite funkciju koja kao parametar prima neki string, a koja transformira taj string u novi string kod kojeg su ASCII šifre svih znakova umanjene za 2 u odnosu na izvorni string (sama funkcija ne vraća *nikakav rezultat*). Na primjer, ukoliko se funkciji proslijedi string "`s`" čiji je sadržaj "`proba`", sadržaj stringa nakon poziva funkcije treba da bude "`npm@?`".

Rješenje (jedno od mogućih):

```
void Transformiraj(string &s) {
  for(int i = 0; i < s.length(); i++) s[i] -= 2;
}
```

Napomena: Formalni parametar "`s`" *mora biti referenca*. Dalje, eksplicitne pretvorbe tipa poput "`s[i] = char(int(s[i]) - 2)`" nisu neispravne, ali *nisu ni potrebne*.

Zadatak 6 (2 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`remove_copy_if`". Ova funkcija kopira blok elemenata između pokazivača *p1* i *p2* na lokaciju određenu pokazivačem *p3*, uz izbacivanje elemenata za koji funkcija *f* vraća kao rezultat "**`true`**" kad joj se proslijede kao argument (i vraća kao rezultat pokazivač koji pokazuje tačno iza posljednjeg elemenata odredišnog bloka), pri čemu je sintaksa poziva ove funkcije "`remove_copy_if(`*p1*`, `*p2*`, `*p3*`, `*f*`)`". Napišite sami generičku funkciju "`Ukloni`" koja prima potpuno iste parametre i obavlja potpuno istu funkciju kao i funkcija "`remove_copy_if`".

```
template <typename NekiTip>
NekiTip *Ukloni(NekiTip *p1, NekiTip *p2, NekiTip *p3, bool f(NekiTip)) {
  for(; p1 < p2; p1++)
    if(!f(*p1)) *p3++ = *p1;
  return p3;
}
```

Napomena: Ovako napisana funkcija koristi *djelimičnu dedukciju tipa*, i stoga se može primijeniti *samo na pokazivače*, a ne i na iteratore. Prava funkcija "`replace_if`" radi i sa iteratorima, što bi se moglo postići *potpunom dedukcijom tipa*, kao u sljedećoj izvedbi (ovdje "`PokazivackiTip`" predstavlja neki tip koji se ponaša *poput pokazivača*, što uključuje i iteratore):

```
template <typename NekiTip, typename PokazivackiTip>
PokazivackiTip Zamijeni(PokazivackiTip p1, PokazivackiTip p2,
  PokazivackiTip p3, bool f(NekiTip)) {
    for(; p < p2; p++)
      if(!f(*p1)) *p3++ = *p1;
    return p3;
}
```

Zadatak 7 (2 poena)

Napišite funkciju koja ima četiri parametra, pri čemu su prva dva parametra niz cijelih brojeva i broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar funkcije smjesti respektivno broj jednocifrenih i broj dvocifrenih elemenata u nizu. Pored toga, funkcija treba da vrati kao rezultat logičku vrijednost "**`true`**" ako i samo ako je makar jedan element niza jednak nuli (inače vraća logičku vrijednost "**`false`**"). Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati samo kako biste pozvali ovu funkciju i ispisali rezultate koje ona donosi na primjeru nekog niza sa fiksno zadanim elementima.

Rješenje (jedno od mogućih):

```
bool JednoDvocifreni(int niz[], int br_el, int &br_jed, int &br_dvo) {
  bool ima_nula(false);
  br_jed = br_dvo = 0;
  for(int i = 0; i < br_el; i++) {
    if(niz[i] >= 0 && niz[i] <= 9) br_jed++;
    else if(niz[i] >= 10 && niz[i] <= 99) br_dvo++;
    else if(niz[i] == 0) ima_nula = true;
  }
  return ima_nula;
}
```

Isječak glavnog programa:

```
int a[5] = {17, 4, 332, 0, 15};
int brj, brd;
bool ima = Cjelobrojni(a, 5, brj, brd);
if(ima) cout << "Ima nula\n";
else cout << "Nema nula\n";
cout << "Jednocifrenih: " << brj << " Dvocifrenih: " << brd << endl;
```

Napomena: Formalni parametri "`br_c`" i "`br_nc`" *moraju biti reference.*

Zadatak 8 (3 poena)

Napišite generičku funkciju koja vrši dinamičku alokaciju prostora za pamćenje kvadratne matrice formata $n \times n$, pri čemu je n jedan od parametara funkcije (elementi matrice mogu biti proizvoljnog tipa). Parametri funkcije su referenca na dvojni pokazivač koji će služiti za pristup tako kreiranom prostoru, kao i dimenzija matrice n. Funkcija ne vraća nikakav rezultat, nego samo pridružuje adresu alociranog prostora pokazivačk koji joj je prosljeđen kao parametar. U slučaju da alokacija ne uspije, funkcija treba da baci tekst "Neuspješna alokacija" kao izuzetak. Također, funkcija se treba pobrinuti da ni u koj slučaju ne može doći do curenja memorije. Napišite i mali isječak glavnog programa (ne treba sve) u kojem ćete demonstrirati kako biste pozvali ovu funkciju, i eventualno uhvatiti izuzetak koji bi mogao biti bačen iz nje.

Rješenje (jedno od mogućih):

```
template <typename TipElemenata>
void KreirajMatricu(TipElemenata **&mat, int n) {
  mat = 0;
  try {
    mat = new TipElemenata*[n];
    for(int i = 0; i < n; i++) mat[i] = 0;
    for(int i = 0; i < n; i++) mat[i] = new TipElemenata[n];
  }
  catch(...) {
    for(int i = 0; i < n; i++) delete[] mat[i];
    delete[] mat;
```

```
    throw "Neuspješna alokacija!";
  }
}
```

Isječak glavnog programa:

```
double **m(0);
try {
  KreirajMatricu(m, 10)
  ...
}
catch(const char poruka[]) {
  cout << poruka << endl;
}
```

Napomena: Vrijede iste napomene kao u rješenju Zadatka 8. u Grupi A.

POPRAVNI ISPIT IZ PREDMETA "TEHNIKE PROGRAMIRANJA" (I PARCIJALNI)

Zadatak 1. (4 poena)

Napišite šta će se prikazati kao rezultat izvršavanja sljedećeg programa:

```
#include <iostream>

using namespace std;

int &funkcija (int &a, int *b, int c) {
  c *= 1024;
  a = *b *a;
  int &ref = *b;
  a++;
  return ref;
}

int main() {
  int jedan, dva, tri;
  jedan = dva = tri = 1;
  tri += ++dva;
  funkcija(jedan, &dva, tri) = 5;
  cout << jedan << endl << dva << endl << tri << endl;
  return 0;
}
```

Zadatak 2. (11 poena)

Napišite program koji će od korisnika tražiti da unosi kompleksne brojeve, završno sa kompleksnim brojem 0. Unesene brojeve treba stavljati u vektor kompleksnih brojeva koji će se kasnije sortirati po apsolutnoj vrijednosti (modulu) u rastući poredak, koristeći ugrađenu funkciju "`sort`" iz biblioteke "`algorithm`" (uz prethodno definiranu funkciju kriterija). Nakon toga, sortirani niz treba kopirati u dva dinamički alocirana niza od kojih će jedan sadržavati realne dijelove, a drugi kompleksne dijelove brojeva iz vektora. Na kraju je potrebno ispisati realni dio onog kompleksnog broja koji ima najmanji modul.

Napomena: Funkcija "`sort`" ima sintaksu "`sort(`*pocetak*`, `*iza_kraja*`, `*kriterij*`)`".

Zadatak 3. (5 poena)

Napišite program koji će napraviti vektor realnih brojeva popunjen kvadratima prvih 100 prirodnih brojeva. Nakon toga, bez upotrebe petlji na mjesto svakog broja u vektoru treba upisati ostatak pri dijeljenju tog broja sa brojem 42, pripremiti vektor za binarno pretraživanje i binarnim pretraživanjem za broj unesen sa tastature ustanoviti da li se nalazi u nizu ili ne.

Napomena: Sintaksa funkcija "`transform`" i "`binary_search`", koje će Vam zatrebati, glasi

`transform(` *pocetak*`, `*iza_kraja*`, `*odrediste*`, `*funkcija*`)`"
`binary_search(` *pocetak*`, `*iza_kraja*, *vrijednost*`)`"