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

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa, uz kratko obrazloženje zbog čega su rezultati onakvi kakvi jesu (prikaz bez obrazloženja biće shvaćen kao prepisivanje i NEĆE BITI PRIHVAĆEN). Oprez: bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;
void P(int *x, int *y) {
  while(x != y) cout << *x++ << " ";
  cout << endl;
}
int &Q(int &x, int &y, int &z) {
  cout << setw(4) << x << y << z << endl;
  x += 2; y += 3; z += 4;
  cout << x << "," << y << "," << z << endl;
  return x;
}
int main() {
  int x[] = {3, 6, 1, 2, 7, 5, 3, 2, 6, 4};
  cout << 2 + complex<int>(x[1], x[3]) << endl;
  P(x + 2, x + 8);
  int *y(x + 6), *z(new int(x[6]));
  Q(x[4], *z, y[-2])--;
  delete z;
  P(y - 5, &x[9]);
  return 0;
}
```

Rješenje:

```
(8,2)
1 2 7 5 3 2
  737
13,6,13
6 1 2 12 5 3 2 6
```

Zadatak 2 (2 poena)

Napišite funkciju koja kao parametar prima neki prirodan broj i koja vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li u broju koji je proslijeđen kao parametar ima jednakih cifara ili ne. Napišite i kratki isječak programa u kojem ćete testirati kako se poziva napisana funkcija i kako se može iskoristiti njen rezultat.

Rješenje:

Jedna od mogućnosti je da se unutar funkcije sve cifre izdvoje u niz ili vektor, nakon čega se problem svodi na poznati problem da li među elementima niza odnosno vektora ima međusobno jednakih elemenata ili ne. Međutim, to je nepotrebni gubitak memorije i nepotrebno usložnjavanje rješenja (mada efektno demonstrira strategiju svođenja novog problema na ranije riješeni problem). Ovdje prikazano rješenje ne koristi nikakav pomoćni niz niti vektor:

```
bool ImaLiJednakihCifara(int broj) {
  while(broj > 0) {
    int cifra(broj % 10), pomocni(broj / 10);
    while(pomocni > 0) {
      if(cifra == pomocni % 10) return true;
      pomocni /= 10;
    }
    broj /= 10;
  }
  return false;
}

...
int n;
cin >> n;
if(ImaLiJednakihCifara(n)) cout << "Medju ciframa ima jednakih\n";
else cout << "Sve su cifre razlicite\n";
```

Zadatak 3 (2 poena)

Za neki element a_i nekog slijeda brojeva kažemo da je *lokalni maksimum* ukoliko je veći od svojih susjeda, tj. ukoliko vrijedi $a_i > a_{i-1}$ i $a_i > a_{i+1}$. Napišite funkciju koja kao parametar prima vektor realnih brojeva, a kao rezultat vraća novi vektor u kojem se nalaze svi lokalni maksimumi vektora koji je proslijeđen kao parametar (odnosno prazan vektor ako lokalnih maksimuma nema). Napišite i kratki isječak programa u kojem ćete testirati kako se poziva napisana funkcija i kako se može iskoristiti njen rezultat.

Rješenje:

```
vector<double> LokalniMaksimumi(const vector<double> &v) {
  vector<double> lmax;
  for(int i = 1; i < v.size() - 1; i++)
    if(v[i] > v[i - 1] && v[i] > v[i + 1]) lmax.push_back(v[i]);
  return lmax;
}

...
vector<double> v;
...
vector<double> v1(LokalniMaksimumi(v));
if(v1.size() == 0) cout << "Nema lokalnih maksimuma!\n";
else {
  cout << "Lokalni maksimumi su:\n";
  for(int i = 0; i < v1.size(); i++) cout << v1[i] << endl;
}
```

Zadatak 4 (2,5 poena)

Napišite generičku funkciju sa 4 parametra. Prvi parametar je niz elemenata proizvoljnog tipa, za koje se pretpostavlja da se mogu međusobno uspoređivati, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar respektivno smjesti najmanji element niza i prvi element niza koji mu slijedi po veličini (tj. u treći i četvrti parametar treba izdvojiti prva dva elementa po veličini). Pri tome, niz *ne smijete sortirati*. U slučaju da ne postoji drugi element po veličini (tj. ukoliko su svi elementi niza jednaki), funkcija treba baciti izuzetak u kojem govori o tome. Napišite i mali isječak programa u kojem ćete demonstrirati kako se može upotrijebiti napisana funkcija (što uključuje i hvatanje eventualno bačenog izuzetka).

Rješenje:

Problem se najlakše rješava u dva prolaza. Naime, nakon što u prvom prolazu pronađemo najmanji element, u drugom prolazu se lako nalazi onaj koji mu slijedi po veličini. Međutim, isto tako lako je i

previdjeti razne specijalne slučajeve koji mogu nastupiti, odnosno lako je napraviti "skoro" korektna rješenja koja rade uvijek, osim u nekim specijalnim slučajevima (stoga će predmetni nastavnik biti dosta tolerantan pri pregledanju ovog zadatka). Ovdje prikazano rješenje radi korektno u svim slučajevima (postoje i kraća rješenja, ali su previše zasnovana na trikovima):

```
template <typename TipElemenata>
void DvaNajmanja(TipElemenata niz[], int broj_elemenata,
  TipElemenata &min, TipElemenata &do_min) {
    min = niz[0];
    for(int i = 1; i < broj_elemenata; i++)
      if(niz[i] < min) min = niz[i];
    bool ima_drugog(false);
    for(int i = 0; i < broj_elemenata; i++)
      if(niz[i] != min) {
        do_min = niz[i]; ima_drugog = true; break;
      }
    if(!ima_drugog) throw "Niz ne sadrzi dva razlicita elementa!\n";
    for(int i = 0; i < broj_elemenata; i++)
      if(niz[i] < do_min && niz[i] != min) do_min = niz[i];
}

...
int test[10] = {3, 5, 1, 6, 9, 1, 7, 9, 6, 4};
try {
  int p, q;
  DvaNajmanja(test, 10, p, q);
  cout << p << " " << q << endl;
}
catch(const char poruka[]) {
  cout << poruka;
}
```

Zadatak 5 (2 poena)

Jedan od načina "šatrovačkog" govora među djecom, koji je popularan u nekim dijelovima naše države, sastoji se u tome da se nakon svakog samoglasnika ubaci slovo "p" i ponovljeni taj samoglasnik. Na primjer, umjesto "dobar dan" kaže se "dopobapar dapan". Napravite funkciju koja kao parametar prima objekat tipa "`string`" za koji se pretpostavlja da predstavlja neku recenicu na našem jeziku, a koja kao rezultat daje novi string koji predstavlja istu rečenicu prevedenu na "šatrovački" u skladu sa opisanim pravilom. Radi jednostavnosti, pretpostavite da rečenica sadrži samo velika slova.

Rješenje:

```
string Satrovacki(const string &s) {
  string s1;
  for(int i = 0; i < s.length(); i++) {
    s1 += s[i];
    if(s[i] == 'A' || s[i] == 'E' || s[i] == 'I' || s[i] == 'O'
      || s[i] == 'U')
        s1 = s1 + 'P' + s[i];
  }
  return s1;
}
```

Neznatno složenije rješenje može se izvesti pomoću operacije "`push_back`".

NAPOMENA: Mada izgleda čudno, naredba "`s1 = s1 + 'P' + s[i]`" ne može se zamijeniti sa na prvi pogled ekvivalentnom naredbom "`s1 += 'P' + s[i]`". Problem je u prioritetu operacija i osjetljivosti jezika C++ na tipove operanada. Naime, u ovako napisanoj naredbi, prvo se izvršava dio izraza "`'P' + s[i]`". U njemu su oba operanda tipa "`char`", pa je i rezultat tipa "`char`" (jedan znak

čija je ASCII šifra jednaka zbiru ASCII šifara operanada) a ne string od dva znaka što bi nam trebalo. Ne radi ispravno čak ni konstrukcija "`s1 += "P" + s[i]`". Naime, operand `"P"` nije tipa "`string`" kako mnogi pogrešno misle nego tipa "**`const char []`**", što se opet dalje pri sabiranju automatski konvertira u "**`const char *`**" i što u konačnici ne vodi ka ničemu dobrom.

Zadatak 6 (2,5 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`replace_copy_if`" sa 5 parametara *p*1, *p*2, *p*3, *f* i *v* koja kopira blok elemenata između pokazivača *p1* i *p2* na lokaciju određenu pokazivačem *p3* uz zamjenu svakog elementa za koji funkcija *f* vraća kao rezultat "**`true`**" kad joj se proslijedi kao argument sa elementom koji ima vrijednost *v*. Napravite generičku funkciju koja prima potpuno iste parametre i koja obavlja potpuno isti zadatak kao i funkcija "`replace_copy_if`" iz biblioteke "`algorithm`" (funkciju nazovite prema vlastitoj volji). Funkcija bi trebala biti zasnovana na potpunoj dedukciji tipova, tako da se umjesto pokazivača mogu koristiti i iteratori (ukoliko ne znate izvesti potpunu dedukciju, koristite parcijalnu dedukciju, samo ćete time izgubiti 1 poen). Napišite i mali isječak programa u kojem ćete demonstrirati kako biste ovu funkciju primijenili da prepišete elemente nekog fiksnog niza od 10 cijelih brojeva u drugi niz uz zamjenu svih parnih brojeva nulama.

Rješenje:

```
template <typename PokTip, typename TipVrijednosti>
void KopirajUzUvjetnuZamjenu(PokTip p1, PokTip p2, PokTip p3,
  bool f(TipVrijednosti), TipVrijednosti v) {
    while(p1 != p2) {
      if(f(*p1)) *p3 = v;
      else *p3 = *p1;
      p1++; p3++;
    }
}

...
bool DaLiJeParan(int broj) {
  return broj % 2 == 0;
}

...
int a[10] = {3, 5, 1, 2, 8, 5, 4, 9, 6, 7};
int b[10];
...
KopirajUzUvjetnuZamjenu(a, a + 10, b, DaLiJeParan, 0);
```

NAPOMENA: Strogo uzevši, bibliotečka funkcija "`replace_copy_if`" vraća kao rezultat pokazivač iza posljednjeg elementa odredišnog bloka. Taj bi se dodatak lako izveo promjenom povratnog tipa funkcije iz "**`void`**" u "`PokTip`" i dodavanjem naredbe "**`return`** `p3`" na kraj. Također, u ovom rješenju ipak je za parametar *f* korištena djelimična dedukcija, jer je navedeno da je taj parametar *funkcija*. Može se koristiti i potpuna dedukcija po ovom parametru što bi omogućilo da se kao parametar *f* ne zadaju samo funkcije nego i funkcijski objekti (funktori). Ovo nije izvedeno zbog činjenice da studenti koji tek slušaju kurs prvi put (brucoši) još ne znaju šta su funkcijski objekti. Zbog toga, ova napomena se odnosi na one koji bi ovo mogli znati, poput studenata sa vrhunskim predavanjem i studenata ponovaca (?).

Zadatak 7 (2 poena)

U nekim primjenama u kompjuterskoj grafici (npr. za konstrukciju poligona sa tjemenima u zadatim tačkama čije se stranice ne sijeku) javlja se potreba za sortiranjem niza kompleksnih brojeva u rastući poredak po argumentima (faznim uglovima). U slučaju da dva kompleksna broja imaju isti argument, prije treba doći onaj čiji je modul manji. Definirajte odgovarajuću funkciju kriterija i napišite isječak programa u kojem ćete pokazati kako biste deklarirali vektor kompleksnih brojeva, napunili ga vrijednostima unesenim sa tastature i sortirali ga u traženi poredak korištenjem funkcije "`sort`" iz

biblioteke “algorithm” (podsjetimo se da ova funkcija prima kao parametre dva pokazivača ili iteratora koji omeđuju blok koji se sortira i opcionalno funkciju kriterija kao treći parametar, dok se argument i modul kompleksnog broja dobijaju pomoću funkcija “arg” i “abs” respektivno).

Rješenje:

```
bool Kriterij(complex<double> z1, complex<double> z2) {
  if(arg(z1) != arg(z2)) return arg(z1) < arg(z2);
  return abs(z1) < abs(z2);
}

...
vector<complex<double> > v(10);            // recimo 10...
for(int i = 0; i < 10; i++) cin >> v[i];
sort(v.begin(), v.end(), Kriterij);
```

Zadatak 8 (2 poena)

Napišite funkciju sa jednim parametrom n koja dinamički alocira prostor za kvadratnu matricu formata $n \times n$ i popunjava sadržaj matrice “tablicom množenja” za brojeve od 1 do n, tj. vrijednost elementa u presjeku i-tog reda i j-te kolone treba da bude $i \cdot j$. Funkcija treba da kao rezultat vrati dvojni pokazivač koji služi za pristup elementima tako kreirane matrice. Alokaciju obavite postupkom kontinualne alokacije. U slučaju da je $n \leq 0$, funkcija treba da baci tekst “Broj elemenata mora biti pozitivan” kao izuzetak. U slučaju da alokacija ne uspije, funkcija treba da baci tekst “Alokacija nije uspjela” kao izuzetak (pri tome treba paziti da ne dođedo curenja memorije). Napisanu funkciju iskoristite u testnom programu u kojem se sa tastature unosi broj n, nakon čega se ispisuju elementi kreirane matrice (tablica množenja). Na kraju se vrši oslobađanje prostora koji je zauzela dinamički alocirana matrica. Pored toga, u glavnom programu treba predvidjeti hvatanje svih izuzetaka koji bi funkcija eventualno mogla baciti.

Rješenje:

```
int **tablica_mnozenja(int n) {
  if(n <= 0) throw "Broj elemenata mora biti pozitivan!\n";
  int **mat(new int*[n]);
  try {
    mat[0] = new int[n * n];
    for(int i = 1; i < n; i++) mat[i] = mat[i - 1] + n;
    for(int i = 0; i < n; i++)
      for(int j = 0; j < n; j++) mat[i][j] = (i + 1) * (j + 1);
  }
  catch(...) {
    delete[] mat;
    throw "Alokacija nije uspjela!\n";
  }
  return mat;
}

...
int n;
cin >> n;
try {
  int **mat(tablica_mnozenja(n));
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) cout << setw(3) << mat[i][j];
    cout << endl;
  }
  delete[] mat[0]; delete[] mat;
}
catch(const char poruka[]) {
  cout << poruka;
}
```

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

Zadatak 1 (5 poena)

Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa, uz kratko obrazloženje zbog čega su rezultati onakvi kakvi jesu (prikaz bez obrazloženja biće shvaćen kao prepisivanje i NEĆE BITI PRIHVAĆEN). Oprez: bitan je svaki razmak, kao i prelasci u nove redove. Radi jasnoće, razmake prikažite kao kvadratiće.

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;
void P(int *x, int *y) {
  while(x != y) cout << *x++ << " ";
  cout << endl;
}
int &Q(int &x, int &y, int &z) {
  cout << setw(4) << x << y << z << endl;
  x += 4; y += 3; z += 2;
  cout << x << "," << y << "," << z << endl;
  return x;
}
int main() {
  int x[] = {4, 5, 2, 1, 8, 4, 4, 1, 7, 3};
  cout << 2 + complex<int>(x[1], x[3]) << endl;
  P(x + 2, x + 8);
  int *y(x + 6), *z(new int(x[6]));
  Q(x[4], *z, y[-2])--;
  delete z;
  P(y - 5, &x[9]);
  return 0;
}
```

Rješenje:

```
(7,1)
2 1 8 4 4 1
   848
14,7,14
5 2 1 13 4 4 1 7
```

Zadatak 2 (2 poena)

Napišite funkciju koja kao parametar prima neki prirodan broj i koja vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li su u broju koji je proslijeđen kao parametar sve cifre različite ili ne. Napišite i kratki isječak programa u kojem ćete testirati kako se poziva napisana funkcija i kako se može iskoristiti njen rezultat.

Rješenje:

Jedna od mogućnosti je da se unutar funkcije sve cifre izdvoje u niz ili vektor, nakon čega se problem svodi na poznati problem da li su svi elementi niza odnosno vektora međusobno različiti ili ne. Međutim, to je nepotrebni gubitak memorije i nepotrebno usložnjavanje rješenja (mada efektno demonstrira strategiju svođenja novog problema na ranije riješeni problem). Ovdje prikazano rješenje ne koristi nikakav pomoćni niz niti vektor:

```
bool DaLiSuSveCifreRazlicite(int broj) {
  while(broj > 0) {
    int cifra(broj % 10), pomocni(broj / 10);
    while(pomocni > 0) {
      if(cifra == pomocni % 10) return false;
      pomocni /= 10;
    }
    broj /= 10;
  }
  return true;
}

...
int n;
cin >> n;
if(DaLiSuSveCifreRazlicite(n)) cout << "Sve su cifre razlicite\n";
else cout << "Medju ciframa ima jednakih\n";
```

Zadatak 3 (2 poena)

Za neki element a_i nekog slijeda brojeva kažemo da je *lokalni minimum* ukoliko je manji od svojih susjeda, tj. ukoliko vrijedi $a_i < a_{i-1}$ i $a_i < a_{i+1}$. Napišite funkciju koja kao parametar prima vektor realnih brojeva, a kao rezultat vraća novi vektor u kojem se nalaze svi lokalni minimumi vektora koji je proslijeđen kao parametar (odnosno prazan vektor ako lokalnih minimuma nema). Napišite i kratki isječak programa u kojem ćete testirati kako se poziva napisana funkcija i kako se može iskoristiti njen rezultat.

Rješenje:

```
vector<double> LokalniMinimumi(const vector<double> &v) {
  vector<double> lmin;
  for(int i = 1; i < v.size() - 1; i++)
    if(v[i] < v[i - 1] && v[i] < v[i + 1]) lmin.push_back(v[i]);
  return lmin;
}

...
vector<double> v;
...
vector<double> v1(LokalniMinimumi(v));
if(v1.size() == 0) cout << "Nema lokalnih maksimuma!\n";
else {
  cout << "Lokalni maksimumi su:\n";
  for(int i = 0; i < v1.size(); i++) cout << v1[i] << endl;
}
```

Zadatak 4 (2,5 poena)

Napišite generičku funkciju sa 4 parametra. Prvi parametar je niz elemenata proizvoljnog tipa, za koje se pretpostavlja da se mogu međusobno uspoređivati, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da u treći i četvrti parametar respektivno smjesti najveći element niza i prvi element niza koji mu prethodi po veličini (tj. u treći i četvrti parametar treba izdvojiti posljednja dva elementa po veličini). Pri tome, niz *ne smijete sortirati*. U slučaju da ne postoji predzadnji element po veličini (tj. ukoliko su svi elementi niza jednaki), funkcija treba baciti izuzetak u kojem govori o tome. Napišite i mali isječak programa u kojem ćete demonstrirati kako se može upotrijebiti napisana funkcija (što uključuje i hvatanje eventualno bačenog izuzetka).

Rješenje:

Problem se najlakše rješava u dva prolaza. Naime, nakon što u prvom prolazu pronađemo najveći element, u drugom prolazu se lako nalazi onaj koji mu prethodi po veličini. Međutim, isto tako lako je

i previdjeti razne specijalne slučajeve koji mogu nastupiti, odnosno lako je napraviti "skoro" korektna rješenja koja rade uvijek, osim u nekim specijalnim slučajevima (stoga će predmetni nastavnik biti dosta tolerantan pri pregledanju ovog zadatka). Ovdje prikazano rješenje radi korektno u svim slučajevima (postoje i kraća rješenja, ali su previše zasnovana na trikovima):

```
template <typename TipElemenata>
void DvaNajveca(TipElemenata niz[], int broj_elemenata,
  TipElemenata &max, TipElemenata &do_max) {
    max = niz[0];
    for(int i = 1; i < broj_elemenata; i++)
      if(niz[i] > max) min = niz[i];
    bool ima_drugog(false);
    for(int i = 0; i < broj_elemenata; i++)
      if(niz[i] != max) {
        do_max = niz[i]; ima_drugog = true; break;
      }
    if(!ima_drugog) throw "Niz ne sadrzi dva razlicita elementa!\n";
    for(int i = 0; i < broj_elemenata; i++)
      if(niz[i] > do_max && niz[i] != max) do_max = niz[i];
}

...
int test[10] = {3, 5, 1, 6, 9, 1, 7, 9, 6, 4};
try {
  int p, q;
  DvaNajveca(test, 10, p, q);
  cout << p << " " << q << endl;
}
catch(const char poruka[]) {
  cout << poruka;
}
```

Zadatak 5 (2 poena)

Jedan od načina "šatrovačkog" govora među djecom, koji je popularan u nekim dijelovima naše države, sastoji se u tome da se nakon svakog samoglasnika ubaci slovo "p" i ponovljeni taj samoglasnik. Na primjer, umjesto "pozdrav svima" kaže se "dopobapar dapan". Napravite funkciju koja kao parametar prima objekat tipa "`string`" za koji se pretpostavlja da predstavlja neku recenicu na našem jeziku, a koja kao rezultat daje novi string koji predstavlja istu rečenicu prevedenu na "šatrovački" u skladu sa opisanim pravilom. Radi jednostavnosti, pretpostavite da rečenica sadrži samo velika slova.

Rješenje:

```
string Satrovacki(const string &s) {
  string s1;
  for(int i = 0; i < s.length(); i++) {
    s1 += s[i];
    if(s[i] == 'A' || s[i] == 'E' || s[i] == 'I' || s[i] == 'O'
      || s[i] == 'U')
        s1 = s1 + 'P' + s[i];
  }
  return s1;
}
```

Neznatno složenije rješenje može se izvesti pomoću operacije "`push_back`".

NAPOMENA: Mada izgleda čudno, naredba "`s1 = s1 + 'P' + s[i]`" ne može se zamijeniti sa na prvi pogled ekvivalentnom naredbom "`s1 += 'P' + s[i]`". Problem je u prioritetu operacija i osjetljivosti jezika C++ na tipove operanada. Naime, u ovako napisanoj naredbi, prvo se izvršava dio izraza "`'P' + s[i]`". U njemu su oba operanda tipa "`char`", pa je i rezultat tipa "`char`" (jedan znak

čija je ASCII šifra jednaka zbiru ASCII šifara operanada) a ne string od dva znaka što bi nam trebalo. Ne radi ispravno čak ni konstrukcija "`s1 += "P" + s[i]`". Naime, operand `"P"` nije tipa "`string`" kako mnogi pogrešno misle nego tipa "`const char []`", što se opet dalje pri sabiranju automatski konvertira u "`const char *`" i što u konačnici ne vodi ka ničemu dobrom.

Zadatak 6 (2,5 poena)

U biblioteci "`algorithm`" nalazi se generička funkcija "`replace_copy_if`" sa 5 parametara *p*1, *p*2, *p*3, *f* i *v* koja kopira blok elemenata između pokazivača *p1* i *p2* na lokaciju određenu pokazivačem *p3* uz zamjenu svakog elementa za koji funkcija *f* vraća kao rezultat "**`true`**" kad joj se proslijedi kao argument sa elementom koji ima vrijednost *v*. Napravite generičku funkciju koja prima potpuno iste parametre i koja obavlja potpuno isti zadatak kao i funkcija "`replace_copy_if`" iz biblioteke "`algorithm`" (funkciju nazovite prema vlastitoj volji). Funkcija bi trebala biti zasnovana na potpunoj dedukciji tipova, tako da se umjesto pokazivača mogu koristiti i iteratori (ukoliko ne znate izvesti potpunu dedukciju, koristite parcijalnu dedukciju, samo ćete time izgubiti 1 poen). Napišite i mali isječak programa u kojem ćete demonstrirati kako biste ovu funkciju primijenili da prepišete elemente nekog fiksnog niza od 10 realnih brojeva u drugi niz uz zamjenu svih negativnih brojeva nulama.

Rješenje:

```
template <typename PokTip, typename TipVrijednosti>
void KopirajUzUvjetnuZamjenu(PokTip p1, PokTip p2, PokTip p3,
  bool f(TipVrijednosti), TipVrijednosti v) {
    while(p1 != p2) {
      if(f(*p1)) *p3 = v;
      else *p3 = *p1;
      p1++; p3++;
    }
}

...
bool DaLiJeNegativan(int broj) {
  return broj < 0;
}

...
int a[10] = {3, 5, 1, 2, 8, 5, 4, 9, 6, 7};
int b[10];
...
KopirajUzUvjetnuZamjenu(a, a + 10, b, DaLiJeNegativan, 0);
```

NAPOMENA: Strogo uzevši, bibliotečka funkcija "`replace_copy_if`" vraća kao rezultat pokazivač iza posljednjeg elementa odredišnog bloka. Taj bi se dodatak lako izveo promjenom povratnog tipa funkcije iz "**`void`**" u "`PokTip`" i dodavanjem naredbe "**`return`** `p3`" na kraj. Također, u ovom rješenju ipak je za parametar *f* korištena djelimična dedukcija, jer je navedeno da je taj parametar *funkcija*. Može se koristiti i potpuna dedukcija po ovom parametru što bi omogućilo da se kao parametar *f* ne zadaju samo funkcije nego i funkcijski objekti (funktori). Ovo nije izvedeno zbog činjenice da studenti koji tek slušaju kurs prvi put (brucoši) još ne znaju šta su funkcijski objekti. Zbog toga, ova napomena se odnosi na one koji bi ovo mogli znati, poput studenata sa vrhunskim predavanjem i studenata ponovaca (?).

Zadatak 7 (2 poena)

U nekim primjenama u kompjuterskoj grafici (npr. za konstrukciju poligona sa tjemenima u zadatim tačkama čije se stranice ne sijeku) javlja se potreba za sortiranjem niza kompleksnih brojeva u opadajući poredak po argumentima (faznim uglovima). U slučaju da dva kompleksna broja imaju isti argument, prije treba doći onaj čiji je modul veći. Definirajte odgovarajuću funkciju kriterija i napišite isječak programa u kojem ćete pokazati kako biste deklarirali vektor kompleksnih brojeva, napunili ga vrijednostima unesenim sa tastature i sortirali ga u traženi poredak korištenjem funkcije "`sort`" iz

biblioteke “algorithm” (podsjetimo se da ova funkcija prima kao parametre dva pokazivača ili iteratora koji omeđuju blok koji se sortira i opcionalno funkciju kriterija kao treći parametar, dok se argument i modul kompleksnog broja dobijaju pomoću funkcija “arg” i “abs” respektivno).

Rješenje:

```
bool Kriterij(complex<double> z1, complex<double> z2) {
  if(arg(z1) != arg(z2)) return arg(z1) > arg(z2);
  return abs(z1) > abs(z2);
}

...
vector<complex<double> > v(10);           // recimo 10...
for(int i = 0; i < 10; i++) cin >> v[i];
sort(v.begin(), v.end(), Kriterij);
```

Zadatak 8 (2 poena)

Napišite funkciju sa jednim parametrom n koja dinamički alocira prostor za kvadratnu matricu formata $n \times n$ i popunjava sadržaj matrice “tablicom množenja” za brojeve od 1 do n, tj. vrijednost elementa u presjeku i-tog reda i j-te kolone treba da bude $i \cdot j$. Funkcija treba da kao rezultat vrati dvojni pokazivač koji služi za pristup elementima tako kreirane matrice. Alokaciju obavite postupkom kontinualne alokacije. U slučaju da je $n \leq 0$, funkcija treba da baci tekst “Broj elemenata mora biti pozitivan” kao izuzetak. U slučaju da alokacija ne uspije, funkcija treba da baci tekst “Alokacija nije uspjela” kao izuzetak (pri tome treba paziti da ne dođedo curenja memorije). Napisanu funkciju iskoristite u testnom programu u kojem se sa tastature unosi broj n, nakon čega se ispisuju elementi kreirane matrice (tablica množenja). Na kraju se vrši oslobađanje prostora koji je zauzela dinamički alocirana matrica. Pored toga, u glavnom programu treba predvidjeti hvatanje svih izuzetaka koji bi funkcija eventualno mogla baciti.

Rješenje:

```
int **tablica_mnozenja(int n) {
  if(n <= 0) throw "Broj elemenata mora biti pozitivan!\n";
  int **mat(new int*[n]);
  try {
    mat[0] = new int[n * n];
    for(int i = 1; i < n; i++) mat[i] = mat[i - 1] + n;
    for(int i = 0; i < n; i++)
      for(int j = 0; j < n; j++) mat[i][j] = (i + 1) * (j + 1);
  }
  catch(...) {
    delete[] mat;
    throw "Alokacija nije uspjela!\n";
  }
  return mat;
}

...
int n;
cin >> n;
try {
  int **mat(tablica_mnozenja(n));
  for(int i = 0; i < n; i++) {
    for(int j = 0; j < n; j++) cout << setw(3) << mat[i][j];
    cout << endl;
  }
  delete[] mat[0]; delete[] mat;
}
catch(const char poruka[]) {
  cout << poruka;
}
```