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

Svi zadaci odnose se na program koji je dat u Prilogu.

Zadatak 1 (1 poen)

Program u Prilogu ima curenje memorije. Napišite sav potreban kod za rješavanje tog problema.

Zadatak 2 (3 poena)

Pretpostavimo da je riješen problem opisan u Zadatku 1. Zbog čega će sada konstrukcije poput

```
Fakultet f;
...
Fakultet f2(f);
...
Fakultet f3;
f3 = f;
```

dovoditi do raznih neželjenih posljedica, poput krahiranja programa, curenja memorije, i sličnih stvari? Napišite sav potreban kod da se eliminiraju sve te nepoželjne posljedice. Pri tome se pobrinite da optimizirate rad ukoliko se pojavi potreba za kopiranjem privremenih objekata tipa "`Fakultet`" (recimo, rezultata neke funkcije koja vraća takve objekte).

Zadatak 3 (2,5 poena)

Analizom koda iz priloga možemo vidjeti da se na naš Fakultet može upisati maksimalno 1000 studenata. Prepravite program tako da je broj studenata koji se mogu upisati na fakultet ograničen samo dostupnom memorijom računara (bez korištenja vektora). U slučaju da se prekorači kapacitet memorije, potrebno je baciti izuzetak tipa "`range_error`" uz prateći tekst "Nema dovoljno memorije", ali pri čemu dalji rad sa Fakultetom mora biti moguć (tj. sam objekat ne smije biti upropašten).

Zadatak 4 (1 poen)

Prepravite program u prilogu tako da je moguće pisati i ovo:

```
Fakultet f(2000);
```

pri čemu se kreira fakultet kapaciteta 2000 studenata. Pri tome, pobrinite se da ne bude moguća automatska konverzija cijelog broja u objekat tipa "`Fakultet`", npr. konstrukcijom tipa

```
f = 2000;          // Ovo ne smije raditi!
```

Deklaracija Fakulteta bez ovog cjelobrojnog parametra treba raditi kao i dosad.

Zadatak 5 (3 poena)

Napravite klasu "`MasterStudent`" čija metoda "`Ispisi`" daje ovakav ispis:

```
Student Meho Mehić ima prosjek 7.5 na prvom ciklusu, a 8 na drugom ciklusu
```

Da bi ovo bilo moguće, potrebno je u ovu klasu dodati i novu metodu "`RegistrirajOcjenuMaster`" analognu metodi "`RegistrirajOcjenu`". Treba biti moguće upisati studenta master studija na isti Fakultet na isti način (pozivom iste metode) kao i običnog studenta. Pored toga, metoda "`Fakultet::IspisiSve`" treba na prikladan način tretirati studente master studija (tj. za njih treba ispisivati prosjek i prvog i drugog ciklusa). Pri tome ćete morati napraviti i dvije sitne izmjene u klasi "`Student`", koje morate navesti. U klasi "`Fakultet`" nije dozvoljeno praviti nikakve izmjene.

Zadatak 6 (1 poen)

Nažalost, inovacije uvedene u Zadatku 5 dovešće do toga da podrška kopiranju riješena u Zadatku 2 neće korektno kopirati studente master studija (prilikom kopiranja, oni će izgubiti svoje specifičnosti). Izvedite neophodne prepravke potrebne da se ovaj problem riješi.

Zadatak 7 (1 poen)

Omogućite da se studenti mogu međusobno porediti po prosjeku, odnosno napišite neophodan kod da bi bile moguće konstrukcije poput sljedećih:

```
Student s1("Pero", "Perić"), s2("Simo", "Simić");
...
if(s1 > s2) std::cout << "Pero ima bolji prosjek";
else std::cout << "Simo ima bolji prosjek";
```

Radi jednostavnosti, podržite samo poređenje pomoću znaka ">". Ne smijete dodavati nikakve nove atribute niti metode ni u jednu klasu.

Zadatak 8 (3 poena)

Vidimo da se u metodi "`Student::RegistrirajOcjenu`" ne radi ništa sa parametrom "`predmet`". Zašto to ne bismo promijenili? Napišite sav potreban kod da bi bile moguće konstrukcije poput

```
Student s("Huso", "Husić");
s.RegistrirajOcjenu("TP", 9);
std::cout << s["TP"] << std::endl;          // Ispisuje 9 (ocjena iz TP-a)
s["TP"] = 10;                               // Popravlja ocjenu iz TP-a na 10
std::cout << s["LAG"] << std::endl;         // Baca izuzetak
```

U posljednjem slučaju (neregistrirana ocjena) treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Ocjena nije registrirana". Također treba obezbijediti da izmjena registrirane ocjene nije moguća za konstantne objekte tipa "`Student`".

Zadatak 9 (2 poena)

Napišite neophodan kod koji je potreban da bi se podaci o studentima upisanim na Fakultet mogli ispisati na ekran ili u datoteku konstrukcijama poput

```
Fakultet f;
...
std::cout << f;
std::ofstream datoteka("FAKULTET.TXT");
datoteka << f;
```

Ispis treba biti u istom formatu kakav se dobija i pozivom metode "`Fakultet::IspisiSve`".

Zadatak 10 (2,5 poena)

Napišite neophodan kod koji omogućava da se podaci o studentima na Fakultetu pročitaju iz tekstualne datoteke. Datoteka je organizirana na sljedeći način. U prvom redu datoteke nalazi se broj studenata. Nakon toga, sljedeći podaci se redom ponavljaju za svakog studenta. Prvo se u jednom redu nalazi znak "B" ili "M" ovisno da li je u pitanju obični student (bachelor) ili student master. Zatim, slijedi razmak pa ime i prezime studenta, razdvojeni razmakom. U sljedećem redu se nalazi broj ocjena, nakon čega u svakom redu slijedi prvo šifra predmeta (jedna riječ), a zatim nakon jednog razmaka i ocjena iz tog predmeta. Ukoliko se radi o studentu master studija, ista takva skupina podataka se ponavlja i za ocjene predmeta sa master studija. Radi jednostavnosti, pretpostavite da datoteka sadrži isključivo ispravne podatke.

Prilog:

```
#include <iostream>
#include <string>
#include <stdexcept>

class Student {
  std::string ime, prezime;
  int ocjene[100];
  int broj_ocjena;
  double DajProsjek() const {
    double suma(0);
    for(int i = 0; i < broj_ocjena; i++) suma += ocjene[i];
    return suma / broj_ocjena;
  }
public:
  Student(std::string ime, std::string prezime) : ime(ime), prezime(prezime),
    broj_ocjena(0) {}
  void RegistrirajOcjenu(std::string predmet, int ocjena) {
    if(ocjena < 5 || ocjena > 10)
      throw std::domain_error("Neispravna ocjena!");
    if(broj_ocjena == 100)
      throw std::range_error("Ne može se registrirati više od 100 ocjena!");
    ocjene[broj_ocjena++] = ocjena;
  }
  void Ispisi() const {
    std::cout << "Student " << ime << " " << prezime << " ima prosjek "
      << DajProsjek();
  }
};

class Fakultet {
  Student **studenti;
  int broj_studenata, kapacitet;
public:
  Fakultet() : studenti(new Student*[1000]), broj_studenata(0),
    kapacitet(1000) {}
  void UpisiStudenta(Student *s) {
    if(broj_studenata == kapacitet)
      throw std::range_error("Popunjen fakultet!");
    studenti[broj_studenata++] = s;
  }
  void IspisiSve() {
    for(int i = 0; i < broj_studenata; i++) {
      studenti[i]->Ispisi();
      std::cout << std::endl;
    }
  }
};

int main() {
  Fakultet f;
  Student *s = new Student("Meho", "Mehić");
  s->RegistrirajOcjenu("IM1", 6);
  s->RegistrirajOcjenu("TP", 9);
  f.UpisiStudenta(s);
  f.IspisiSve();
}
```

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

Svi zadaci odnose se na program koji je dat u Prilogu.

Zadatak 1 (1 poen)

Program u Prilogu ima curenje memorije. Napišite sav potreban kod za rješavanje tog problema.

Zadatak 2 (3 poena)

Pretpostavimo da je riješen problem opisan u Zadatku 1. Zbog čega će sada konstrukcije poput

```
Preduzece p;
...
Preduzece p2(p);
...
Preduzece p3;
p3 = p;
```

dovoditi do raznih neželjenih posljedica, poput krahiranja programa, curenja memorije, i sličnih stvari? Napišite sav potreban kod da se eliminiraju sve te nepoželjne posljedice. Pri tome se pobrinite da optimizirate rad ukoliko se pojavi potreba za kopiranjem privremenih objekata tipa "`Preduzece`" (recimo, rezultata neke funkcije koja vraća takve objekte).

Zadatak 3 (2,5 poena)

Analizom koda iz priloga možemo vidjeti da se u našem Preduzeću može zaposliti maksimalno 1000 radnika. Prepravite program tako da je broj radnika koji se mogu zaposliti u preduzeću ograničen samo dostupnom memorijom računara (bez korištenja vektora). U slučaju da se prekorači kapacitet memorije, potrebno je baciti izuzetak tipa "`range_error`" uz prateći tekst "Nema dovoljno memorije", ali pri čemu dalji rad sa Preduzećem mora biti moguć (tj. sam objekat ne smije biti upropašten).

Zadatak 4 (1 poen)

Prepravite program u prilogu tako da je moguće pisati i ovo:

```
Preduzece p(2000);
```

pri čemu se kreira preduzeće kapaciteta 2000 radnika. Pri tome, pobrinite se da ne bude moguća automatska konverzija cijelog broja u objekat tipa "`Preduzece`", npr. konstrukcijom tipa

```
p = 2000;          // Ovo ne smije raditi!
```

Deklaracija Preduzeća bez ovog cjelobrojnog parametra treba raditi kao i dosad.

Zadatak 5 (3 poena)

Napravite klasu "`Rukovodilac`" čija metoda "`Ispisi`" daje ovakav ispis:

```
Uposlenik Meho Mehić ima prosjek plate 1200 KM redovno, i jos 300 KM ostvareno
na osnovu dodatnih aktivnosti
```

Da bi ovo bilo moguće, potrebno je u ovu klasu dodati i novu metodu "`RegistrirajDodatne`" analognu metodi "`RegistrirajPlatu`". Treba biti moguće zaposliti rukovodioca u isto Preduzeće na isti način (pozivom iste metode) kao i običnog radnika. Pored toga, metoda "`Preduzece::IspisiSve`" treba na prikladan način tretirati rukovodioce (tj. za njih treba ispisivati i prosjek redovne plate i dodatnih aktivnosti). Pri tome ćete morati napraviti i dvije sitne izmjene u klasi "`Radnik`", koje morate navesti. U klasi "`Preduzece`" nije dozvoljeno praviti nikakve izmjene.

Zadatak 6 (1 poen)

Nažalost, inovacije uvedene u Zadatku 5 dovešće do toga da podrška kopiranju riješena u Zadatku 2 neće korektno kopirati rukovodioce (prilikom kopiranja, oni će izgubiti svoje specifičnosti). Izvedite neophodne prepravke potrebne da se ovaj problem riješi.

Zadatak 7 (1 poen)

Omogućite da se radnici mogu međusobno porediti po prosječnoj plati, odnosno napišite neophodan kod da bi bile moguće konstrukcije poput sljedećih:

```
Radnik r1("Pero", "Perić"), r2("Simo", "Simić");
...
if(r1 > r2) std::cout << "Pero ima bolju platu";
else std::cout << "Simo ima bolju platu";
```

Radi jednostavnosti, podržite samo poređenje pomoću znaka ">". Ne smijete dodavati nikakve nove atribute niti metode ni u jednu klasu.

Zadatak 8 (3 poena)

Vidimo da se u metodi "`Radnik::RegistrirajPlatu`" ne radi ništa sa parametrom "`posao`". Zašto to ne bismo promijenili? Svrha ovog parametra trebala bi da bude da se za svaku registriranu platu može voditi evidencija o poslu koji je radnik radio da zaradi tu platu. Napišite sav potreban kod da bi bile moguće konstrukcije poput

```
Radnik r("Huso", "Husić");
r.RegistrirajPlatu("zidanje", 600);
std::cout << r["zidanje"] << std::endl;     // Ispisuje 600 (plata za zidanje)
r["zidanje"] = 700;                         // Popravlja platu za zidanje
std::cout << r["kopanje"] << std::endl;     // Baca izuzetak
```

U posljednjem slučaju (neregistriran posao) treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Posao nije registriran". Također treba obezbijediti da izmjena registrirane plate nije moguća za konstantne objekte tipa "`Radnik`".

Zadatak 9 (2 poena)

Napišite neophodan kod koji je potreban da bi se podaci o radnicima zaposlenim u Preduzeću mogli ispisati na ekran ili u datoteku konstrukcijama poput

```
Preduzece p;
...
std::cout << p;
std::ofstream datoteka("PREDUZECE.TXT");
datoteka << p;
```

Ispis treba biti u istom formatu kakav se dobija i pozivom metode "`Preduzece::IspisiSve`".

Zadatak 10 (2,5 poena)

Napišite neophodan kod koji omogućava da se podaci o radnicima u Preduzeću pročitaju iz tekstualne datoteke. Datoteka je organizirana na sljedeći način. U prvom redu datoteke nalazi se broj uposlenika. Nakon toga, sljedeći podaci se redom ponavljaju za svakog uposlenika. Prvo se u jednom redu nalazi znak "R" ili "U" ovisno da li je u pitanju obični radnik ili rukovodilac (član uprave). Zatim, slijedi razmak pa ime i prezime uposlenika, razdvojeni razmakom. U sljedećem redu se nalazi broj registriranih plata, nakon čega u svakom redu slijedi prvo šifra posla (jedna riječ), a zatim nakon jednog razmaka i plata za taj posao. Ukoliko se radi o rukovodiocu, ista takva skupina podataka se ponavlja i za naknade za dodatne aktivnosti. Radi jednostavnosti, pretpostavite da datoteka sadrži isključivo ispravne podatke.

Prilog:

```
#include <iostream>
#include <string>
#include <stdexcept>

class Radnik {
  std::string ime, prezime;
  int plate[100];
  int broj_plata;
  double DajProsjecnuPlatu() const {
    double suma(0);
    for(int i = 0; i < broj_plata; i++) suma += plate[i];
    return suma / broj_plata;
  }
public:
  Radnik(std::string ime, std::string prezime) : ime(ime), prezime(prezime),
    broj_plata(0) {}
  void RegistrirajPlatu(std::string posao, int plata) {
    if(plata < 0 || plata > 2000)
      throw std::domain_error("Neispravna plata!");
    if(broj_plata == 100)
      throw std::range_error("Ne može se registrirati više od 100 plata!");
    plate[broj_plata++] = plata;
  }
  void Ispisi() const {
    std::cout << "Uposlenik " << ime << " " << prezime << " ima prosjek plate "
      << DajProsjecnuPlatu() << " KM";
  }
};

class Preduzece {
  Radnik **radnici;
  int broj_radnika, kapacitet;
public:
  Preduzece() : radnici(new Radnik*[1000]), broj_radnika(0),
    kapacitet(1000) {}
  void ZaposliRadnika(Radnik *r) {
    if(broj_radnika == kapacitet)
      throw std::range_error("Popunjeno preduzece!");
    radnici[broj_radnika++] = r;
  }
  void IspisiSve() {
    for(int i = 0; i < broj_radnika; i++) {
      radnici[i]->Ispisi();
      std::cout << std::endl;
    }
  }
};

int main() {
  Preduzece p;
  Radnik *r = new Radnik("Meho", "Mehić");
  r->RegistrirajPlatu("zidanje", 700);
  r->RegistrirajPlatu("farbanje", 550);
  p.ZaposliRadnika(r);
  p.IspisiSve();
}
```