Ime i prezime: _____ Matični broj: _____

Ispitna pitanja, grupa A

1. [**2 bod.**] Definirajte klasu koja predstavlja sliku na računalu s atributima za širinu i visinu u pikselima te brojem bajta (engl. *byte*) po pikselu. Omogućite postavljanje i dohvaćanje stanja iz objekata klase.

2. [**3 bod.**] Napišite konstruktor tako da omogućite stvaranje objekata klase iz zadatka 1 kako je prikazano izlistanjem kôda:

```
Image lena(1024, 1024, 3), anna;
```

3. [**2 bod.**] Napišite metodu tako da omogućite skaliranje slike iz zadatka 1 prema faktoru skaliranja *sc* tako da se skaliraju sve dimenzije slike. Za $sc < 0.1$, slika se ne mijenja.

4. [**3 bod.**] Napišite metodu tako da omogućite izračun prostora u kB, potrebnog za pohranu slike iz zadatka 1. Napišite metodu tako da omogućite i izračun prostora uz predan postotak kompresije slike $10.0 \leq p \leq 90.0$. Ako je p van raspona, smatra se da je 0.

```
std::cout << "Size:" << lena.getSizeKB() << ", compressed: " << lena.getSizeKB(compressPercentage) << std::endl;
```

5. [**4 bod.**] Napišite funkciju koja za polje objekata iz zadatka 1, predano joj kao argument, provjerava je li najveću sliku među njima moguće pohraniti ako je dostupna količina slobodnog prostora u kilobajtima zadana dodatnim argumentom funkcije. Podrazumijeva se postojanje metode `getSizeKB()` iz zadatka 4. Testirajte napisanu funkciju za polje imena `values`, duljine 10 uz slobodnih 13588kB.

---

6. [**2 bod.**] Ako je dana klasa prema izlistanju k(̂o)da:

```
class PriceHistory {
double *mPrices;
int mCount;
};
```

napišite konstruktor koji omogućuje stvaranje objekta na način

```
PriceHistory ikeaSofa(10, 3420.0); // 10 cijena u povijesti, sve postavljene na 3420.0
```

7. [**4 bod.**] Za klasu `PriceHistory` u zadatku 6 napišite sve potrebno da bi u potpunosti ispravno radila funkcija `run()`. Smatra se da konstruktor iz zadatka 6 postoji.

```
void run() {
PriceHistory ikeaSofa(10, 3420.0);
PriceHistory* primaSofa = new PriceHistory(ikeaSofa);
delete primaSofa;
}
```

8. [**3 bod.**] Za klasu `PriceHistory` iz zadatka 6 napišite sve potrebno da bi u potpunosti ispravno radio kôd dan izlistanjem (smatra se da konstruktor iz zadatka 6 postoji, usporedba se obavlja po prosječnoj cijeni):

```
PriceHistory appleStock(6, 74545), googleStock(10, 41656);
if (appleStock < googleStock) std::cout << appleStock << " - Invest in apple.";
else std::cout << googleStock << " - Invest in google.";
```

Koristiti programski jezik C++, osim gdje je naznačeno. U potpunosti rukovati svim resursima. Poštovati pravila enkapsulacije. Rabiti inicijalizacijske liste konstruktora. Izbjeći bespotrebno kopiranje objekata. Osigurati rad s konstantnim objektima gdje je to moguće. Obavezno potpisati i predati ovaj i sve ostale listove. Ispit se piše 90 minuta.

9. **[3 bod.]** Za klasu `PriceHistory` iz zadatka 6 napišite metodu koja vraća trend kretanja cijena. Ako je između dvaju susjednih cijena unutar povijesti ukupno bilo više padova nego skokova, vraća -1, ako je bilo više skokova vraća 1, a kôd neutralnog trenda vraća 0. Primjerice, za:
[322.6, 333.45, 335.45, 338.22, 318.50] Rezultat je 1
[322.6, 313.45, 315.45, 310.02, 302.15] Rezultat je -1
[322.6, 313.45, 315.45, 315.45, 315.45] Rezultat je 0

10. **[3 bod.]** Za klasu `PriceHistory` iz zadatka 6 napište funkciju koja omogućuje pohranu objekta poslanog joj kao argument u datoteku imena određenog dodatnim argumentom. Svaki put se sadržaj nadodaje na kraj datoteke. Pretpostaviti postojanje `toString()` metode koja vraća `std::string` reprezentaciju objekta. Testirajte funkciju, ime datoteke neka bude Vaše prezime.

---

11. **[2 bod.]** Za danu klasu `Sensor` napišite odgovarajući parametarski konstruktor, metodu `getCurrentRead()` za dohvaćanje trenutnog mjerenja koja generira i vraća nasumičnu vrijednost $0.0 \leq r \leq 10.0$ te metodu koja uključuje, odnosno isključuje senzor ovisno o trenutnom stanju. Unutar objekta održavati prethodno stanje.

```
class Sensor {
  std::string mName;
  bool mIsActive;
  double mPreviousRead;
}

Sensor accelerationSensor("Acceleration");
```

12. **[4 bod.]** Iz klase `Sensor` dane u zadatku 11 izvedite klasu `LoggingSensor` koja kao atribut ima pokazivač na objekt klase `ConsoleLogger` i prilikom svakog očitanja korištenjem metode `getCurrentRead()` bilježi poruku na konzolu uporabom `ConsoleLogger`-a u obliku:

```
ConsoleLogger logger;
LoggingSensor accelerationSensor("Acceleration", &logger);
accelerationSensor.getCurrentRead();
```

Izlaz:

```
Acceleration sensor. Current: 9.85. Past: 9.63. // Brojevi su nasumicni, vec prema zadanoj funkcionalnosti.
```

Pretpostaviti ispravno napisan parametarski konstruktor klase `Sensor`. Napisati sve potrebno (klase, metode, funkcije...) kako bi se k(o)d dan u zadatku mogao u potpunosti ispravno izvesti.

Koristiti programski jezik C++, osim gdje je naznačeno. U potpunosti rukovati svim resursima. Poštovati pravila enkapsulacije. Rabiti inicijalizacijske liste konstruktora. Izbjeći bespotrebno kopiranje objekata. Osigurati rad s konstantnim objektima gdje je to moguće. Obavezno potpisati i predati ovaj i sve ostale listove. Ispit se piše 90 minuta.