ASP – Sortovi - Zadaci za vježbu

Zadano je polje brojeva s elementima: **6, 1, 10, 3, 2, 8, 4, 7, 9, 5**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **bubble sort**.

6	1	10	3	2	8	4	7	9	5
1	**6**	10	3	2	8	4	7	9	5
1	6	**3**	**10**	2	8	4	7	9	5
1	6	3	**10**	**2**	8	4	7	9	5
1	6	3	2	**10**	**8**	4	7	9	5
1	6	3	2	8	**10**	**4**	7	9	5
1	6	3	2	8	4	**10**	**7**	9	5
1	6	3	2	8	4	7	**10**	**9**	5
1	6	3	2	8	4	7	9	**10**	**5**
1	**6**	**3**	2	8	4	7	9	5	10
1	3	**6**	**2**	8	4	7	9	5	10
1	3	2	6	**8**	**4**	7	9	5	10
1	3	2	6	4	**8**	**7**	9	5	10
1	3	2	6	4	7	8	**9**	**5**	10
1	**3**	**2**	6	4	7	8	5	9	10
1	2	3	**6**	**4**	7	8	5	9	10
1	2	3	4	6	7	**8**	**5**	9	10
1	2	3	4	6	**7**	**5**	8	9	10
1	2	3	4	**6**	**5**	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

Zadano je polje brojeva s elementima: **6, 1, 10, 3, 2, 8, 4, 7, 9, 5**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **insertion sort**.

6	**1**	10	3	2	8	4	7	9	5
1	6	10	3	2	8	4	7	9	5
1	**6**	**10**	**3**	2	8	4	7	9	5
1	**3**	**6**	**10**	**2**	8	4	7	9	5
1	2	3	6	**10**	**8**	4	7	9	5
1	2	3	**6**	**8**	**10**	**4**	7	9	5
1	2	3	4	6	**8**	**10**	**7**	9	5
1	2	3	4	6	7	8	**10**	**9**	5
1	2	3	4	**6**	**7**	**8**	**9**	**10**	**5**
1	2	3	4	5	6	7	8	9	10

Zadano je polje brojeva s elementima: **6, 1, 10, 3, 2, 8, 4, 7, 9, 5**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **shellsort** za niz koraka

a) $h_k = \{4, 3, 1\}$
b) $h_k = \{3, 2, 1\}$

a)

6	1	10	3	**2**	8	4	7	9	5	
2	1	**10**	3	6	8	**4**	7	9	5	
2	1	4	3	6	**8**	10	7	9	**5**	
2	1	4	3	6	5	**10**	7	9	**8**	h_k=4
2	**1**	4	3	6	5	8	7	9	10	h_k=3
1	2	**4**	**3**	6	5	8	7	9	10	
1	2	3	4	**6**	**5**	8	7	9	10	
1	2	3	4	5	6	**8**	**7**	9	10	
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	h_k=1

b)

6	1	10	**3**	2	8	4	7	9	5	
3	1	**10**	6	2	**8**	4	7	9	5	
3	1	8	**6**	2	10	**4**	7	9	5	
3	1	8	4	2	**10**	6	7	**9**	5	
3	1	8	4	2	9	**6**	7	10	**5**	
3	1	**8**	4	**2**	9	5	7	10	6	h_k=3
2	1	3	4	**8**	9	**5**	7	10	6	
2	1	3	4	5	**9**	8	**7**	10	6	
2	1	3	4	5	**7**	8	**9**	10	**6**	
2	**1**	3	4	5	6	8	7	10	9	h_k=2
1	2	3	4	5	6	**8**	**7**	10	9	
1	2	3	4	5	6	7	8	**10**	**9**	
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	h_k=1

Zadano je polje brojeva s elementima: **6, 1, 10, 3, 2, 8, 4, 7, 9, 5**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **quicksort** u kojem se stožerni element:

a) određuje metodom aproksimacije medijana korištenjem 3 člana niza.

6	1	10	3	2	8	4	7	9	5
5	1	10	3	2	6	4	7	9	8
5	1	**10**	3	2	9	**4**	7	6	8
5	1	4	3	2	**9**	10	7	**6**	8
5	1	4	3	2	6	10	7	9	8
5	1	4	3	2	6	10	7	9	8
2	1	4	3	5	6	8	7	9	10
2	1	3	4	5	6	8	7	**9**	10
2	1	3	**4**	5	6	**8**	**7**	9	10
2	1	3	4	5	6	7	8	9	10
2	1	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

b) uzima prvi element

6	1	**10**	3	2	8	4	7	9	**5**
6	1	5	3	2	**8**	**4**	7	9	10
6	1	5	3	2	4	8	7	9	10
6	1	5	3	2	**4**	8	7	9	10
4	1	5	3	2	6	8	7	9	10
4	1	**5**	3	**2**	6	8	7	9	10
4	1	2	3	5	6	**8**	**7**	9	10
4	1	2	**3**	5	6	7	8	9	10
3	1	2	4	5	6	7	8	9	10
3	1	2	4	5	6	7	8	9	10
3	1	**2**	4	5	6	7	8	9	10
2	1	3	4	5	6	7	8	9	10
2	1	3	4	5	6	7	8	9	10
2	**1**	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

c) uzima posljednji element

6	1	10	3	2	8	**4**	7	9	5
4	1	**10**	3	**2**	8	6	7	9	5
4	1	2	3	10	8	6	7	9	5
4	1	2	3	**10**	8	6	7	9	**5**
4	1	2	3	5	8	6	7	9	10
4	1	2	3	5	8	6	7	9	10
4	1	**2**	3	5	8	6	7	9	10
2	1	4	3	5	8	6	7	9	10
2	1	**4**	**3**	5	8	6	7	9	10
2	1	3	4	5	6	8	7	9	10
2	**1**	3	4	5	6	8	7	9	10
1	2	3	4	5	6	**8**	**7**	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

Zadaci za samostalnu vježbu:

1. Zadano je polje brojeva s elementima: **10, 2, 3, 15, 7, 16, 5, 6, 17, 8**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **bubble sort**.

2. Zadano je polje brojeva s elementima: **10, 2, 3, 15, 7, 16, 5, 6, 17, 8**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **insertion sort**.

3. Zadano je polje brojeva s elementima: **10, 2, 3, 15, 7, 16, 5, 6, 17, 8**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **shellsort** za niz koraka
   a) $h_k = \{4, 3, 1\}$
   b) $h_k = \{3, 2, 1\}$

1. Zadano je polje brojeva s elementima: **10, 2, 3, 15, 7, 16, 5, 6, 17, 8**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog niza brojeva algoritmom **quicksort** u kojem se stožerni element:
   a) određuje metodom aproksimacije medijana korištenjem 3 člana niza.
   b) uzima prvi element
   c) uzima posljednji element

ASP – Stog - Zadaci za vježbu

1. Na stog realiziranom poljem, spremaju se podaci tipa float.
Napisati funkcije za rad sa stogom (skidanje sa stoga i stavljenje na stog).
Kada je stog pun, funkcija za stavljanje novog elementa na stog treba pokušati dinamički
povećati veličinu stoga za 10% trenutne veličine stoga.

```
#include <stdio.h>
#include <stdlib.h>

float* skini(float *stog, int *vrh, float *element) {
    if (*vrh < 0) return NULL;
    *element  = stog[*vrh];
    (*vrh)--;
    return stog;
}

float* stavi(float *stog, int *vrh, int *n, float element) {
    if (*vrh >= (*n)-1) {
        stog = realloc(stog, (1.1*(*n))*sizeof(float));
        *n *= 1.1;
    }
    *vrh += 1;
    stog[*vrh] = element;
    return stog;
}

void main() {
    float *stog;
    int n=10; int vrh;
    float element;
    int retval;
   int i = 0;

   stog = (float*) malloc(sizeof(float)*n);
   vrh = -1;

   for(i=0; i<20; i++)
   {
         printf("Stavljam %d\n", i);
         stog = stavi(stog, &vrh, &n, i);
   }

   while(stog)
   {
         stog = skini(stog, &vrh, &element);
         if(stog != NULL)
               printf("Skidam %f\n", element);
   }
}
```

2. Na stog realiziran jednostruko povezanom listom spremaju se cjelobrojni podaci (**long**). Napisati funkcije za stavljanje novog elementa na stog i skidanje elementa s vrha stoga.

```
// dodaje element na vrh stoga
atom *dodaj (atom *vrh, tip element) {
     atom *novi; // pokazivac na novi atom
     if ((novi = (atom *) malloc(sizeof(atom))) != NULL) {
          novi->element = element;
          novi->sljed = vrh;
          printf("Na adresu %p dodao sam %d, a sljedeci je
%p\n",

     novi, element, vrh);
  }
  return novi; // vrati pokazivac na novi atom
}
// skida element s vrha stoga
atom *skini (atom *vrh, tip *element) {
     atom *pom;                    // pomocni pokazivac
        if (vrh == NULL) return NULL;
     *element = vrh->element;
     printf ("S adrese %p ", vrh);
     pom = vrh->sljed;   // sacuvaj novi vrh
     free (vrh);                   // oslobodi vrh
     return pom;                   // vrati novi vrh
}
```

ASP – OOP - Zadaci za vježbu

1. Implementirati konstruktore, destruktor i metode klase `Skup` koja predstavlja skup elemenata tipa float:

```
class Skup
{
private:
   float *_elementi;
   int _brojElemenata;
public:
   Skup();
   Skup(int brojElemenata);
   void Dodaj(float element);
   float BrojElemenata();
   int BrojPojavljivanja(float element);
      ~Skup();
};
```

   Defaultni konstruktor alocira memoriju za 10 elemenata, postavlja broj elemenata na 0. Drugi konstruktor alocira memoriju za zadani broj elemenata (i postavlja broj elemenata na 0).
   Metoda `Dodaj` dodaje element na kraj do sada popunjenog niza elemenata, metoda `BrojElemenata` vraća broj elemenata, a metoda `BrojPojavljivanja` vraća koliko se puta zadani element pojavio u skupu.

```
class Skup
{
private:
        float *_elementi;
        int _brojElemenata;
        int _maxBrojElemenata;
public:
        Skup();
        Skup(int brojElemenata);
        void Dodaj(float element);
        float BrojElemenata();
        int BrojPojavljivanja(float element);
~Skup();
};

Skup::Skup()
{
        _elementi = new float[10];
        _maxBrojElemenata = 10;
        _brojElemenata = 0;
}
Skup::Skup(int brojElemenata)
```

```
{
	_elementi = new float[brojElemenata];
	_maxBrojElemenata = brojElemenata;
	_brojElemenata = 0;
}
void Skup::Dodaj(float element)
{
	_elementi[_brojElemenata] = element;
	_brojElemenata++;
}
float Skup::BrojElemenata()
{
	return _brojElemenata;
}
int Skup::BrojPojavljivanja(float element)
{
	int br = 0;
	for (int i = 0; i < _brojElemenata; i++)
		if (_elementi[i] == element) br++;
	return br;
}
Skup::~Skup()
{
 delete [] _elementi;
}
```

2. Zadane su klase Kruznica i Tocka sa sljedećim definicijama:

```
class Kruznica
{
public:
   Kruznica() {}
   Kruznica(float inX, float inY, float inRad)
   {
         _x = inX; _y = inY; _radius = inRad;
   }

   float GetCenterX() { return _x; }
   float GetCenterY() { return _y; }
   float GetRadius()  { return _radius; }

private:
   float       _x, _y;
   float       _radius;
};

class Tocka
{
public:
   Tocka() {}
   Tocka(float inX, float inY)
   {
         _x = inX; _y = inY;
   }

   float       GetX() { return _x; }
   float       GetY() { return _y; }

private:
   float       _x, _y;

};
```

Potrebno je napisati funkciju koja će za zadanu kružnicu i zadano polje točaka, prebrojati koliko se točaka nalazi unutar zadane kružnice. Prototip funkcije je:

```
int KolikoUnutar(Kruznica &kruz, Tocka poljeTocaka[], int n);
```

```
int     KolikoUnutar(Kruznica &kruz, Tocka poljeTocaka[], int n)
{
        int             Num = 0;
        for( int i=0; i<n; i++ )
        {
                double          x1 = kruz.GetCenterX();
                double          y1 = kruz.GetCenterY();
                double          x2 = poljeTocaka[i].GetX();
                double          y2 = poljeTocaka[i].GetY();

                double          Dist = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1)*(x2 - y1));

                if( Dist < kruz.GetRadius() )
                        Num++;
```

```
        }

        return Num;
}
```

Zadaci za vježbu su navedeni u posebnom dokumentu (2MI-Vjezba.doc), a može ih se pronaći i u sklopu programa s predavanja iz OOP-a.

Ovaj dokument nije službeni. Namjera mu je dati uvid u zadatke koji bi se MOGLI naći na međuispitu. Moguće je da će se u međuispitu javiti i lakši i teži zadaci od navedenih.

SORTOVI

2. Zadan je isti niz brojeva brojeva **6, 7, 2, 1, 10, 4, 9, 5, 3, 8**.. Brojevi su zapisani u polje.

a) *(30 bodova)* Ilustrirati uzlazno sortiranje zadanog niza brojeva shell sort-om sa nizom koraka h_k={4, 3, 1}. Ispisati izgled polja nakon svake zamjene dvaju brojeva.

b) *(30 bodova)* Ilustrirati uzlazno sortiranje zadanog niza brojeva sortom umetanjem (insertion sort). Ispisati izgled polja nakon svake zamjene dvaju brojeva.

1. Zadan je niz brojeva: **10**, **3**, **4**, **7**, **1**, **2**, **12**, **6**, **11**, **5**, **9**, **8**. Ilustrirati uzlazno (od manjeg prema većem) sortiranje zadanog niza brojeva postupkom ***Shellsort***.
   a) Za korake {5, 3, 1}
   b) Za korake {4, 2, 1}

   Napomena: obavezno ispisati sadržaj niza (polja) nakon zamjene svaka dva broja.

1. Zadano je polje brojeva s elementima: **10, 2, 3, 15, 7, 16, 5, 6, 17, 8**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog polja algoritmom **quicksort** ako se kao stožer odabire zadnji element.

Napomena: obavezno ispisati sadržaj niza (polja) nakon zamjene svaka dva broja.

4. Napisati funkciju koja će zadano polje cijelih brojeva (**int**) sortirati algoritmom **bubble sort** (bilo kojom varijantom algoritma). Funkcija treba imati prototip:

```
void bubble(int *polje, int N);
```

4. Zadano je polje brojeva s elementima: **6, 1, 10, 3, 2, 8, 4, 7, 9, 5**. Ilustrirati (ispisati polje nakon svake promjene) sortiranje zadanog polja algoritmom **quicksort**:

a) ako se kao stožer odabire prvi element

b) ako se stožer odabire metodom medijana

2. Na nizu brojeva: 14, 4, 96, 56, 3, 88, 22, 222, 5, 1 ilustrirati korake **silaznog** quicksorta (ispisati izgled polja prilikom svake zamjene elemenata) kada se kao stožer odabire **prvi** element.
3. **a) (2 boda)** Sljedećim je programskim odsječkom prikazan sort s umetanjem (*engl. insertion sort*). Na označena je mjesta potrebno upisati dijelove koji nedostaju (__) da bi algoritam sortiranja ispravno radio:

```
1.    void InsertionSort (tip A [], int N) {
2.    int i, j;
3.    tip pom;
4.    for (i = 0; i < N-1; i++) {
5.          pom = A[__];
```

```
6.          for (j = __  ; j >= __ && A[j-1] > pom; j--)
7.                A[j] = A[j-1];
8.          A[__] = pom;
9.          }
10.   }
```

Napomena: Na papir koji se predaje ne treba prepisivati cijeli odsječak, već samo dijelove koji nedostaju (navesti i broj retka u kojem nedostaju).

b) (3 boda) Na nizu brojeva: 33, 1, 22, 2, 11, 17, 3, 6 ilustrirati korake **uzlaznog** sorta selekcijom (*engl. selection sort*) (ispisati izgled polja prilikom svake zamjene elemenata).

3. Ilustrirati korake **uzlaznog quicksorta** za podatke: **10, 7, 8, 6, 1, 4, 3, 5, 2, 9**. Stožer odabrati metodom **medijana**. Ispisati izgled polja nakon svake zamjene dvaju elemenata polja.

3. Napisati funkciju koja će zadano polje cijelih brojeva uzlazno sortirati algoritmom **bubble sort**:

a) koristeći osnovnu varijantu algoritma

b) koristeći poboljšanu varijantu algoritma

Obje funkcije trebaju imati prototip:

```
void bubble(int polje[], int n);
```

STOG

1. (*50 bodova*) Na stog realiziran poljem spremaju se cjelobrojni podaci (**long**). Napisati funkciju za stavljanje novog elementa na stog i funkciju za skidanje elementa sa vrha stoga. Odrediti apriornu složenost obje funkcije.

3. (*60 bodova*) Na stog realiziran jednostruko povezanom listom spremaju se cjelobrojni podaci (**long**). Napisati funkcije za stavljanje novog elementa na stog i skidanje elementa s vrha stoga. Napisati dodatnu funkciju koja će učitavati cijele brojeve sve dok se ne učita broj 0 te će ih nakon toga ispisati obrnutim redoslijedom od onog kojim su učitavani. Dodatnu funkciju OBAVEZNO treba realizirati koristeći stog, tj. prije napisane funkcije.

2. (60 bodova) Stog realiziran poljem i stog realiziran listom sadrže podatke tipa **double**. Napisati funkciju za stavljanje elementa na stog realiziran jednostruko povezanom listom i funkciju za skidanje elementa sa stoga realiziranog poljem.

Koristeći te dvije funkcije napisati treću, **rekurzivnu** funkciju, koja će prepisati sve elemente iz stoga realiziranog poljem u stog realiziran listom tako da se očuva redoslijed elemenata (nakon obavljanja treće funkcije stog realiziran listom ostaje prazan). Treća funkcija elementima stogova ne smije pristupati direktno, već samo preko prve dvije funkcije. Stogovi se trećoj funkciji zadaju preko argumenata.

5. (50 bodova) Na stog realiziran poljem spremaju se podaci cjelobrojnog tipa (**long**). Napisati funkcije za skidanje sa stoga i stavljanje na stog. Napisati dodatnu funkciju koja će iz zadanog stoga izbaciti sve nule. Ta funkcija stogu smije pristupati samo preko funkcija za rad sa stogom (ne smije direktno pristupati elementima polja). Dozvoljeno je korištenje pomoćnog stoga.

1. Napisati funkciju koja će korištenjem pomoćnog stoga izbaciti sa stoga sve stavke koje su negativni brojevi. Nakon obavljanja funkcije na stogu trebaju ostati nenegativni brojevi. Funkcija treba vratiti broj izbačenih elemenata.
   Koristiti funkcije `dodaj` i `skini` zadanih prototipova (pretpostaviti da funkcije postoje):
   ```
   int dodaj(float stavka, float stog[], int n, int *vrh);

   int skini(float *stavka, float stog[], int *vrh);
   ```
   Funkcije `dodaj` i `skini` vraćaju 1, ako su odgovarajuće operacije uspješno obavljene, a 0 inače.
2. Stog je realiziran kao jednostruko povezana lista koja sadrži zapise o proizvodima: **šifru proizvoda** (`int`), **naziv proizvoda** (50 + 1 znak), te **cijenu proizvoda** (`float`). Vrh stoga pokazuje na prvi zapis u listi. Potrebno je napisati funkciju koja dodaje novi zapis na stog. Funkcija treba vratiti 1, ako je zapis dodan na stog, a inače 0.
3. Na stog realiziran **jednostruko** povezanom listom spremaju se cjelobrojni podaci (**long**). Napisati funkciju za skidanje jednog podatka sa stoga i funkciju za stavljanje novog podatka na stog. Napisati glavni program koji će učitavati cijele brojeve s tipkovnice sve dok se ne učita broj 0 te ih stavljati na stog. Nakon toga brojeve treba skidati sa stoga i ispisivati (tako da budu ispisani obrnutim redoslijedom od onoga kojim su učitavani).

4. Na stog realiziran poljem spremaju se realni podaci (**double**). Napisati slijedeće:

a) funkciju za skidanje elementa sa stoga

b) funkciju za stavljanje elementa na stog, ako na stogu nema dovoljno mjesta za novi element, funkcija stog treba proširiti za 10 novih mjesta (i zapisati novi element)

c) odsječak glavnog programa u kojem će se deklarirati sve varijable potrebne za rad stoga i zauzeti dovoljno memorijskog prostora da bi se u stog moglo zapisati 10 elemenata.

2. Na stog realiziranom poljem, spremaju se podaci tipa **float**. Napisati funkcije za rad sa stogom (skidanje sa stoga i stavljenje na stog). Kada je stog pun, funkcija za stavljanje novog elementa na stog treba pokušati dinamički povećati veličinu stoga za 10% trenutne veličine stoga. Svaka funkcija treba vratiti 1 ako je operacija uspjela, a 0 inače.

4. Stog realiziran poljem sadrži cjelobrojne podatke (**long**). Napisati funkcije za stavljanje elementa na stog i skidanje elementa sa stoga. Ako na stogu nema mjesta za novi element, stog treba dinamički povećati za 10 novih mjesta (i upisati novi element). Funkcije trebaju vratiti 1 ako je operacija uspjela, a 0 inače.

OOP

Za sve donje zadatke napraviti i podrazumijevani konstruktor, barem jedan konstruktor s parametrima, kopirajući konstruktor.

1.KREIRAJ KLASU KREDIT ČIJE SU VARIJABLE KOLIČINA I KAMATA ODREĐENOG KREDITA(SVI PODACI TIPA DOUBLE), A METODE SU JOJ PRIRAST(KOLICINA *KAMATA) I POVEĆANJE ((PRIRAST+KOLIČINA)/KOLIČINA)

3.KREIRAJ KLASU ROMB ČIJE SU VARIJEBLE STRANICE ROMBA a I DULJINE STRANICA NJEGOVIH DIJAGONALA e i f.(PODACI TIPA DOUBLE),A METODE SU JOJ OPSEG ROMBA,I POVRŠINA ROMBA.

5.KREIRAJ KLASU PRAVOKUTNIK ČIJA SU SVOJSTVA DULJINA STRANICE a i b (podaci tipa int),a metode su joj površina i duljina dijagonale.