

Algoritmi i strukture podataka

2. međuispit

1. (4 bodova) Za zadani niz brojeva: **3, 2, 8, 6, 4, 10, 2, 1, 5, 7, 9**, ilustrirati sortiranje postupkom *quicksort* tako da se kao stožer odabire prvi element u polju. Rješenje mora sadržavati izgled polja nakon svake zamjene dvaju elemenata.

Jedan način

```

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

Drugi način (razlika je u zasjenjenom retku)

```

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

2. (4 bodova) Za zadani niz brojeva: **3, 2, 8, 6, 4, 10, 2, 1, 5, 7, 9**, ilustrirati sortiranje postupkom *shellsort* sa nizom koraka {4,2,1}. Rješenje mora sadržavati izgled polja nakon svake zamjene dvaju elemenata.

```

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

3. (7 bodova) Na stog realiziran poljem spremaju se podaci o točkama iz koordinatnog sustava. Svaka točka predstavljena je sljedećom strukturom:

```
struct tocka{
    float x, y;
}
```

Napisati funkciju za stavljanje točke na stog i funkciju za skidanje točke sa stoga. Obje funkcije trebaju vratiti 1 ako su uspješno obavile posao, a 0 inače.

Također, napisati rekurzivnu funkciju koja će korištenjem gore napisanih funkcija, bez korištenja pomoćnih polja ili stogova, ispisati sadržaj stoga na način da se prvo ispiše točka koja se nalazi na dnu stoga, a tek na kraju točka koja se nalazila na vrhu stoga. Nakon završetka rekurzivne funkcije, poredak točaka na stogu mora ostati nepromijenjen.

```
int skini(struct tocka *stavka, struct tocka stog[], int *vrh){
    if (*vrh < 0) return 0;
    *stavka = stog[*vrh];
    (*vrh)--;
    return 1;
}

int dodaj (struct tocka stavka, struct tocka stog[], int n, int *vrh) {
    if (*vrh >= n-1) return 0;
    (*vrh)++;
    stog[*vrh] = stavka;
    return 1;
}

void ispis(struct tocka stog[], int n, int *vrh){
    struct tocka t;
    if (skini(&t, stog, vrh)){
        ispis(stog, n, vrh);
        printf("%.2f %.2f\n", t.x , t.y);
        dodaj(t, stog, n, vrh);
    }
}
```

4. (5 bodova) Za spremanje cijelih brojeva na stog definirana je klasa `Stog` koja ima jedan konstruktor i tri javne funkcije:

```
Stog::Stog();  
void Stog::Stavi(int element);  
void Stog::Skini(int *element);  
int Stog::Prazan();
```

Funkcija `Prazan` vraća 1 ukoliko je stog prazan, a 0 inače.

Napisati funkciju koja će stvoriti novi stog koji će sadržavati sve one brojeve iz zadanog stoga koji su veći ili jednaki prosječnoj vrijednosti elemenata iz zadanog stoga. Poredak tih brojeva u novom stogu nije bitan. Redoslijed podataka na zadanom stogu mora ostati nepromijenjen. Funkcija mora imati prototip:

```
Stog *veci_od_prosjeka(Stog *zadani);
```

Napomena: Za klasu `Stog` nije definiran copy konstruktor. Možete pretpostaviti da u svakom objektu klase `Stog` ima dovoljno mjesta za dodavanje novih elemenata za stog. Smijete koristiti pomoćne stogove.

```
Stog *veci_od_prosjeka(Stog *zadani){  
    Stog *novi = new Stog(), pom;  
    int suma=0, el, br=0; float prosjek;  
    while(!zadani->Prazan()){  
        zadani->Skini(&el);  
        pom.Stavi(el);  
        suma+=el; br++;  
    }  
    if (br>0)  
        prosjek = (float) suma / br;  
    while(!pom.Prazan()){  
        pom.Skini(&el);  
        zadani->Stavi(el);  
        if (el >= prosjek)  
            novi->Stavi(el);  
    }  
    printf("Prosjek = %.2f", prosjek);  
    return novi;  
}
```