Drugi međuispit iz Algoritama i struktura podataka 16.5.2006.

Napomene za sve zadatke:

- Nije dopušteno korištenje goto naredbe, te statičkih i globalnih varijabli.
- U svakom zadatku u kojem se koristi struktura, strukturu treba i definirati
- **Studentima na pitanja ne smiju odgovarati asistenti koji ih čuvaju u učionici. Na pitanja će odgovarati isključivo asistenti koji za vrijeme ispita obilaze prostorije**

1. Zadan je niz brojeva: **5**, **4**, **9**, **1**, **7**, **10**, **8**, **2**, **3**, **6**. Ilustrirati sortiranje zadanog niza brojeva postupkom ***quicksor****t* tako da se kao stože odabire:

a) prvi element u polju
b) zadnji element u polju

Obavezno ispisati sadržaj polja nakon svake zamjene dvaju elemenata. Nije dopušteno koristiti drugi algoritam za polja manja od neke zadane veličine (*cutoff*, npr. 3 ili 5).

a) **5** – stožer
5 – element koji se zamjenjuje

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

b) **5** – stožer
5 – element koji se zamjenjuje

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

2. Napisati funkciju koja će od dva zadana **sortirana** niza cijelih brojeva, napraviti treći **sortirani** niz koji sadrži sve elemente iz prva dva niza. Dva zadana niza moraju ostati nepromijenjena. Memoriju za treći niz potrebno je zauzeti u funkciji i preko imena (**return**) vratiti u glavni program. Funkcija treba imati prototip:

```
int *merge(int *niz1, int N1, int *niz2, int N2, int *NMerge);
```

Preko argumenta `NMerge`, u glavni program treba vratiti broj elemenata trećeg niza (N1 + N2).

*Primjer: Neka je prvi niz: **1**, **3**, **5**, **6**, a drugi niz: **2**, **4**, **5**, **7**. Treći niz mora biti: **1**, **2**, **3**, **4**, **5**, **5**, **6**, **7** i preko varijable `NMerge` treba vratiti vrijednost 8.*

```
int *merge(int *niz1, int N1, int *niz2, int N2, int *NMerge) {
    int i,j,k;
    int *niz3;

    i = j = k = 0;
    *NMerge = N1 + N2;

    niz3 = (int *)malloc(*NMerge * sizeof(int));
    if (niz3 == NULL) return NULL;

    while (i<N1 && j<N2) {
        if (niz1[i] < niz2[j]) {
            niz3[k] = niz1[i];
            i++;
            k++;
        }
        else {
            niz3[k] = niz2[j];
            j++;
            k++;
        }
    }

    while (i<N1) {
        niz3[k] = niz1[i];
        i++;
        k++;
    }

    while (j<N2) {
        niz3[k] = niz2[j];
        j++;
        k++;

   }

    return niz3;
}
```

3. Na stog realiziran poljem spremaju se cjelobrojni podaci (**int**). Napisati funkciju za stavljanje elementa na stog, i funkciju za skidanje elementa sa stoga. Obje funkcije trebaju vratiti 1 ako su uspješno obavile posao, a 0 inače.

Napisati i glavni program u kojem se sa tipkovnice učitava 10 brojeva i stavlja ih se na stog koristeći prije napisanu funkciju.

```
int stavi(int *stog, int *vrh, int N, int element) {
    if (*vrh >= N-1) return 0;

    (*vrh)++;
    stog[*vrh] = element;
    return 1;
}

int skini(int *stog, int *vrh, int *element) {
    if (*vrh < 0) return 0;

    *element = stog[*vrh];
    (*vrh)--;
    return 1;
}

void main() {
    int stog[20];
    int vrh = -1;
    int element;

    for (int i=0; i<10; i++) {
        scanf("%d", &element);
        stavi(stog, &vrh, 20, element);
    }

    printf("\n\n Elementi stoga:");
    while (skini(stog, &vrh, &element)) {
        printf(" %d ", element);
    }
}
```

4. Za spremanje cijelih brojeva na stog definirana je klasa **`Stog`** koja ima jedan konstruktor i dvije javne funkcije:

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

Funkcije `Skini` i `Stavi` vraćaju 1 ako je operacija uspješno obavljena, a 0 inače.

Napisati funkciju koja će napraviti i vratiti duplikat zadanog stoga (isti takav stog, isti elementi koji imaju isti redoslijed). Zadani stog mora na kraju ostati nepromijenjen. Funkcija mora imati prototip:

```
Stog *duplikat(Stog *zadani);
```

Napomene:
- *dopušteno je koristiti pomoćni stog*
- *možete pretpostaviti da u svakom objektu klase* `Stog` *ima dovoljno mjesta*
- *za klasu* `Stog` *NIJE definiran copy konstruktor*

```
Stog *duplikat(Stog *zadani) {
    Stog *dup = new Stog;
    Stog pom;
    int element;

    while(zadani->Skini(&element)) {
        pom.Stavi(element);
    }

    while (pom.Skini(&element)) {
        dup->Stavi(element);
        zadani->Stavi(element);
    }

    return dup;
}
```