Napomena

U svim problemima koji slijede nije dozvoljena upotreba komandi i funkcija koje dosad nisu korištene na predavanjima ili vježbama. Dozvoljena je upotreba kontejnera iz standardne biblioteke (std::vector i std::list), kao i C nizova.

Za svaki od implementiranih sort algoritama, u komentaru kod implementacije odgovoriti na sljedeća pitanja: - koja je vremenska složenost tog sort algoritma, - nad kojim tipom iteratora implementirani algoritam radi (forward, bidirekcioni ili random access), - da li algoritam koristi dodatnu memoriju ili je in-place, - da li je algoritam stabilan.

Zadatak 1

Implementirati algoritam sortiranja bubble sort sa potpisom:

```
template <typename It, typename C>
void bubblesort(It begin, It end, C comp)
```

Ovaj algoritam treba sortirati sekvencu [begin, end) koristeći pristup bubble sorta. Treći argument je relacija poređenja, comp, i predstavlja funkciju sa potpisom `bool(const T&,const T&)` koja govori kako treba sortirati niz.

Primjera radi, moguće je sortirati niz silazno koristeći poziv:

```
bubblesort(v.begin(), v.end(),
    [](const int& lhs, const int& rhs) { return lhs > rhs; });
```

Zadatak 2

Implementirati algoritam sortiranja selection sort sa potpisom:

```
template <typename It>
void selectionsort(It begin, It end, C comp)
```

Zadatak 3

Implementirati algoritam insertion sort sa potpisom:

```
template <typename It>
void insertionsort(It begin, It end, C comp)
```

Zadatak 4

Implementirati algoritam shell sort sa potpisom:

```
template <typename It>
void shellsort(It begin, It end, C comp)
```

Zadatak 5

Implementirati algoritam sortiranja merge sort metodom sa potpisom:

```
template <typename It>
void mergesort(It begin, It end)
```

Za razliku od implementacije koja je rađena na vježbama, ova implementacija algoritma treba biti in-place, odnosno trebali bi postupak `merge`-anja dva podniza odraditi bez korištenja pomoćnog niza.

Zadatak 6

Implementirati algoritam sortiranja quick sort metodom sa potpisom:

```
template <typename It>
void quicksort(It begin, It end)
```

Za razliku od implementacije koja je rađena na vježbama, ova implementacija treba da, nakon što rekurzijom smanjimo podniz na veličinu 16 ili manje, pozvati insertionsort umjesto quicksorta.

Zadatak 7

Neka je data struktura Tim definisana na sljedeći način:

```
struct Tim {
  std::string  naziv;
  int bodovi;
  int primljeniGolovi;
  int postignutiGolovi;
};
```

Napisati funkciju koja sortira niz objekata klase Tim proizvoljne dužine tako da budu zadovoljeni sljedeći kriteriji:

- timovi na kraju treba da budu sortirani po broju bodova u rastućem redoslijedu,
- ukoliko postoji više timova sa istim brojem bodova, bolje pozicionirani tim je tim sa boljom gol-razlikom (razlikom postignutih i primljenih golova),
- ukoliko postoji više timova sa istom gol-razlikom, bolje pozicioniran je tim sa većim brojem postignutih golova,
- ukoliko postoji dva ili više timova sa istim brojem postignutih i primljenih golova te istim brojem bodova, u konačnom poretku treba da budu međusobno poredani po rastućem abecednom redoslijedu.

Za riješavanje ovog zadatka potrebno je pozivati sort funkcije sa različitim kriterijima sortiranja. Odabrane sort funkcije implementirati tako da primaju funkciju comp kao parametar.

Implementirani algoritam treba biti in-place i imati minimalnu moguću složenost. Objasniti implementirano rješenje.

Način predavanja

Zadaću je potrebno predati u obliku arhive *ime_prezime_zadaca2.zip*, pri čemu sadržaj arhive treba da bude direktorij *ime_prezime_zadaca2*, sa poddirektorijem za svaki zadatak, sa imenima: zadatak1, zadatak2, ..., zadatak7.

Pored toga, potrebno je da se studenti strogo pridržavaju naziva metoda, njihovih parametara i povratnih vrijednosti u zadacima gdje su oni naglašeni. Za izlazak na provjeru potrebno je implementirati i predati minimalno 50% zadaće.

Sve predane zadaće će biti automatski pregledane za plagijate i svaki pokušaj prepisivanja će biti prijavljen i adekvatno sankcionisan.