**Ime Prezime: Grupa B**

1. Opisati dva algoritma koji broje koliko niz sadrži parova brojeva (*a*, -*a*) (npr. *{ 6, 1, -1, -6, 2 }* sadrži dva para). Niz nema više od po jednog para za svaku vrijednost. Prvi algoritam ne sortira niz, a drugi algoritam se oslanja na sortiranje niza (sortira se sa qsort). Za svaki algoritam odrediti vremensku složenost O u ovisnosti o dužini niza N i na temelju O složenosti pojedinih operacija (npr. provjeri da li je -*a* u nizu linearnom pretragom).
2. Za iduće funkcije odrediti T(n) i O(f(n)), gdje je n dužina niza:

int zbroj(int \*niz, int n) {

int z = 0;

for (int i = n; i > 0; i--) {

for(int j=n; j > 0; j=j/2) {

z += niz[j];

}

}

return z;

}

void ispis(int \*niz, int n) {

for (int i = 0; i < n; i++) {

for (int j = i; j < n; j++)

printf("%d ", niz [j]);

for (int j = 0; j < i; j++)

printf("%d ", niz [j]);

}

}

1. Prikazati mergesort i quicksort na idućem nizu: { 5, 3, 9, 6, 2, 5, 2, 7, 7 }.

Za iduće zadatke sa listama je definirana iduća struktura:

typedef struct Element {

int broj;

struct Element \*next;

} Element;

**U svakom zadatku naznačiti da li se koristi „dummy“ head element.**

1. Napisati funkciju koja prima listu i „vraća“ dvije liste preko parametara. Dvije nove liste nastaju od prve liste tako da u jednu idu svi neparni brojevi, a u drugu svi parni brojevi. Npr. za listu {1, 3, 6, 3, 2, 4}, funkcija vraća liste {1, 3, 3} i { 6, 2, 4 } (redoslijed u novim listama je nebitan).

void podijeli(Element \*lst, Element \*\*parni, Element \*\*neparni) {

\*parni = \*neparni = NULL;

while(lst != NULL) {

Element \*tmp = lst;

lst = lst->next;

if (tmp -> broj % 2 == 0) {

tmp->next = \*parni;

\*parni = tmp;

} else {

tmp->next = \*neparni;

\*neparni = tmp;

}

}

}

1. Napisati funkciju koja prima listu i provjerava da li je lista sortirana. Funkcija vraća 1 ako su svi brojevi sortirani uzlazno, -1 ako su svi sortirani silazno ili 0 za sve druge slučajeve.

Int provjera(Element \*lst) {

If (lst == NULL || lst->next == NULL)

Return 0;

If(lst->broj > lst->next->broj) {

While(lst->next != NULL) {

If(lst->broj < lst->next->broj)

Return 0;

}

Return 1;

}

If(lst->broj < lst->next->broj) {

While(lst->next != NULL) {

If(lst->broj > lst->next->broj)

Return 0;

}

Return -1;

}

}