Programski jezik I

– Primjeri sa 11. termina predavanja –

Liste, drveta

1. Elementi liste (cijeli brojevi) upisani u čvorovima liste su sortirani u rastući redosljed. Napisati funkciju kojoj se prosljeđuju pokazivač na listu i cijeli broj, kojeg treba smjestiti u novi član liste postavljen tako da lista ostane sortirana. Funkcija vraća pokazivač na glavu liste.

   Rješenje:

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

typedef struct lista
{
        int vr;
        struct lista *sljed;
} lista;

lista *ubaciCvor(lista *, int);
void stampajListu(lista *);


int main()
{
        lista *glava = NULL;

        glava = ubaciCvor(glava, 5);
        glava = ubaciCvor(glava, 10);
        glava = ubaciCvor(glava, 7);
        glava = ubaciCvor(glava, 3);

        stampajListu(glava);

        return 0;
}

void stampajListu(lista *q)
{
        while (q != NULL)
        {
                printf("%d ", q->vr);
                q = q->sljed;
        }
        printf("\n");
}
```

```c
lista *ubaciCvor(lista *glava, int vrijednost)
{
        lista *novi = (lista *)malloc(sizeof(lista));
        novi->vr = vrijednost;
        novi->sljed = NULL;

        // Ako je lista prazna ili treba ubaciti na početak
        if (glava == NULL || vrijednost < glava->vr)
        {
                novi->sljed = glava;
                return novi;
        }

        // Pronalazak mjesta za umetanje
        lista *trenutni = glava;
        while (trenutni->sljed != NULL && trenutni->sljed->vr < vrijednost)
        {
                trenutni = trenutni->sljed;
        }

        // Umetanje novog čvora između trenutni i trenutni->sljed
        novi->sljed = trenutni->sljed;
        trenutni->sljed = novi;

        return glava;
}
```

2. Elementi liste (cijeli brojevi) upisani su u čvorovima liste. Napisati funkciju kojoj se prosljeđuje glava liste i cio broj. Funkcija treba da izbriše sve čvorove u kojima je upisan broj koji se proslijeđen kao drugi argument funkcije i da vrati glavu liste.

```
01 |          #include <stdio.h>
02 |          #include <stdlib.h>
03 |
04 |          struct lista
05 |          {
06 |                  int vr;
07 |                  struct lista *sljed;
08 |          };
09 |
10 |          void stampajListu(struct lista *);
11 |          struct lista *brisiCvor(struct lista *, int);
12 |
13 |          int main()
14 |          {
15 |                  struct lista *p, *q, *t;
16 |                  int x;
17 |                  printf("Unijeti elemente liste (lista se zavrsava unosom
       prvog broja veceg od 50): ");
18 |                  p = (struct lista *)malloc(sizeof(struct lista));
19 |                  p->sljed = NULL;
20 |                  scanf("%d", &x);
21 |                  p->vr = x;
22 |                  t = p;
23 |                  while (1)
24 |                  {
25 |                          q = (struct lista *)malloc(sizeof(struct lista));
26 |                          scanf("%d", &x);
27 |                          if (x > 50)
28 |                          {
29 |                                  break;
30 |                          }
31 |                          q->sljed = NULL;
32 |                          q->vr = x;
33 |                          t->sljed = q;
34 |                          t = q;
35 |                  }
36 |                  stampajListu(p);
37 |
38 |                  p = brisiCvor(p, 4);
39 |
40 |                  stampajListu(p);
41 |                  return 0;
42 |                  /**/
43 |          }
44 |
45 |          void stampajListu(struct lista *q)
46 |          {
47 |                  while (q != NULL)
48 |                  {
49 |                          printf("%d ", q->vr);
50 |                          q = q->sljed;
51 |                  }
52 |                  printf("\n");
53 |          }
```

```
01 |          struct lista *brisiCvor(struct lista *glava, int value)
02 |          {
03 |                  // Ako je lista prazna, samo vratite glavu
04 |                  if (glava == NULL)
05 |                  {
06 |                          return glava;
07 |                  }
08 |
09 |                  // Pomoćni čvor za praćenje trenutnog čvora i prethodnog č
       vora
10 |                  struct lista *trenutni = glava;
11 |                  struct lista *prethodni = NULL;
12 |
13 |                  // Iterirajte kroz listu
14 |                  while (trenutni != NULL)
15 |                  {
16 |                          // Ako je trenutni čvor čvor koji treba izbrisati
17 |                          if (trenutni->vr == value)
18 |                          {
19 |                                  // Ako je čvor koji se briše prvi čvor
20 |                                  if (prethodni == NULL)
21 |                                  {
22 |                                          struct lista *temp = trenutni;
23 |                                          glava = trenutni->sljed;
24 |                                          trenutni = glava;
25 |                                          free(temp);
26 |                                  }
27 |                                  else
28 |                                  {
29 |                                          prethodni->sljed = trenutni->sljed;
30 |                                          free(trenutni);
31 |                                          trenutni = prethodni->sljed;
32 |                                  }
33 |                          }
34 |                          else
35 |                          {
36 |                                  // Pomaknite se na sljedeći čvor
37 |                                  prethodni = trenutni;
38 |                                  trenutni = trenutni->sljed;
39 |                          }
40 |                  }
41 |
42 |                  return glava;
43 |          }
44 |
```

3. Formirati binarno drvo u čijem svakom čvorovima je upisan cio broj. Napisati funkcije za štampanje/obilazak drveta inorder pravilom. Napisati i funkciju koja štampa visinu drveta.

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

struct drvo
{
        int i;
        struct drvo *left;
        struct drvo *right;
};

void stampajInorder(struct drvo *);
int visina(struct drvo *);

int main()
{
        struct drvo *p = NULL, *q = NULL, *r = NULL, *t = NULL;

        // 1
        p = (struct drvo *)malloc(sizeof(struct drvo));
        if (p == NULL)
                exit(1);
        p->i = 1;
        p->left = NULL;
        p->right = NULL;

        // 2
        q = (struct drvo *)malloc(sizeof(struct drvo));
        if (q == NULL)
                exit(1);
        q->i = 2;
        q->left = NULL;
        q->right = NULL;
        p->left = q;

        // 3
        r = (struct drvo *)malloc(sizeof(struct drvo));
        if (r == NULL)
                exit(1);
        r->i = 3;
        r->left = NULL;
        r->right = NULL;
        p->right = r;

        t = q;

        // 4
        q = (struct drvo *)malloc(sizeof(struct drvo));
        if (q == NULL)
                exit(1);
        q->i = 4;
        q->left = NULL;
        q->right = NULL;
        t->left = q;

        // 5
        q = (struct drvo *)malloc(sizeof(struct drvo));
        if (q == NULL)
                exit(1);
        q->i = 5;
        q->left = NULL;
```

```
61 |          q->right = NULL;
62 |          t->right = q;
63 |
64 |          t = q;
65 |
66 |          // 6
67 |          q = (struct drvo *)malloc(sizeof(struct drvo));
68 |          if (q == NULL)
69 |                  exit(1);
70 |          q->i = 6;
71 |          q->left = NULL;
72 |          q->right = NULL;
73 |          t->left = q;
74 |
75 |          // 7
76 |          q = (struct drvo *)malloc(sizeof(struct drvo));
77 |          if (q == NULL)
78 |                  exit(1);
79 |          q->i = 7;
80 |          q->left = NULL;
81 |          q->right = NULL;
82 |          r->right = q;
83 |
84 |          int x = visina(p);
85 |          printf("Visina drveta je %d", x);
86 |  }
87 |
88 |  void stampajInorder(struct drvo *root)
89 |  {
90 |          if (root != NULL)
91 |          {
92 |                  stampajInorder(root->left);
93 |                  printf("%d ", root->i);
94 |                  stampajInorder(root->right);
95 |          }
96 |  }
97 |
98 |  int visina(struct drvo *root)
99 |  {
100 |         if (root == NULL)
101 |         {
102 |                 return 0;
103 |         }
104 |         else
105 |         {
106 |                 // Rekurzivno izračunavanje visine lijevog i desnog
    podstabla
107 |                 int visinaLijevo = visina(root->left);
108 |                 int visinaDesno = visina(root->right);
109 |
110 |                 // Visina stabla je maksimalna visina između lijevog i
    desnog podstabla, plus 1 za korijen
111 |                 return (visinaLijevo > visinaDesno ? visinaLijevo :
    visinaDesno) + 1;
112 |         }
113 |  }
```