Strukture podataka i algoritmi 1

Test 2024-25 – max 26 poena

07.09.2025.

1. **[2 poena]** Napisati rezultate sledećih kodova.

a)
```
int a = 5, b = 2;
int *p = &a;
*p = *p + b;
b = *p * 2;
printf("%d %d %d\n", a, b,
*p);
```
7 14 7

b)
```
#define KVADRAT(x) (x * x)

int main() {
   int a = 3;
   int b = KVADRAT(a + 1);
   printf("%d\n", b);
return 0;
}
```
7

c)
```
char c = 'A';
putchar(c);
putchar(c + 3);
putchar('\n');
```
AD

d)
```
int a = 5, b = 10;
int *p = &a, *q = &b;
*p = *q;
*q = *p + 5;
printf("%d %d\n", a, b);
```
10 15

2. a) **[1 poen]** Objasniti razliku između cp i mv.

cp pravi kopiju fajla/direktorijuma, dok mv premesti fajl/direktorijum ili ga preimenuje.

b) **[1 poen]** Koja je razlika između komandi:
gcc file.c -o program
gcc file.c –c

gcc file.c -o program pravi direktno izvršni fajl program.
gcc file.c -c pravi samo objektni fajl file.o.

3. **[1 poen]** Šta je rezultat sledećeg koda?

```
#include <stdio.h>

int main() {
   int a = 6, b = 3;
   int r = (a & b) ^ 4;
   printf("r = %d\n", r);
   return 0;
}
```
r = 6

4. **[1.5 poen]** Koji će biti izlaz programa?

```
#include <stdio.h>

void fun(int arr[], int start, int end) {
   if (start >= end) return;
   arr[start] = arr[start] + arr[end];
   arr[end] = arr[start] - arr[end];
   arr[start] = arr[start] - arr[end];
   fun(arr, start+1, end-1);
}

int main() {
   int arr[] = {5, 8, 6, 9};
   int n = sizeof(arr)/sizeof(arr[0]);
   fun(arr, 0, n-1);
   for (int i=0; i<n; i++) {
      printf("%d ", arr[i]);
   }
   return 0;
}
```

9 6 8 5

5. **[1.5 poen]** Šta je rezultat sledećeg koda?

```
#include <stdio.h>

int fun(int x) {
   static int s = 5;
   if (x % 2 == 0)
      s += x;
   else
      s -= x;
   return s;
}

int main() {
   printf("%d\n", fun(3));
   printf("%d\n", fun(4));
   printf("%d\n", fun(5));
   return 0;
}
```

2
6
1

6. **[1.5 poen]** Napiši funkciju koja proverava da li je k-ti bit broja x postavljen na 1.
Ulaz: ceo broj x, pozicija k.
Izlaz: ispis "BIT JE POSTAVLJEN" ili "BIT NIJE POSTAVLJEN".

```
void proveri_bit(int x, int k) {
   if (x & (1 << k))
      printf("BIT JE POSTAVLJEN\n");
   else
      printf("BIT NIJE POSTAVLJEN\n");
}
```

7. **[2 poena]** Napiši program koji:
Dinamički alocira niz od n celih brojeva (korisnik unosi n).
Popunjava niz unosom sa tastature.
Za svaki element niza:
   - Ako je paran → postavi poslednji bit na 1.
   - Ako je neparan → obriši poslednji bit (postavi na 0).

Ispiši novi niz.

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

int main() {
   int n, i;
   printf("Unesi n: ");
   scanf("%d", &n);

   int *a = (int *)malloc(n * sizeof(int));
   if (!a) {
      printf("Nema dovoljno memorije!\n");
      return 1;
   }

   printf("Unesi elemente niza:\n");
   for (i = 0; i < n; i++) {
      scanf("%d", &a[i]);
   }

   for (i = 0; i < n; i++) {
      if (a[i] % 2 == 0)
         a[i] |= 1;   // postavi poslednji bit
      else
         a[i] &= ~1;  // obriši poslednji bit
   }

   printf("Izmenjen niz: ");
   for (i = 0; i < n; i++) {
      printf("%d ", a[i]);
   }
   printf("\n");

   free(a);
   return 0;
}
```

8. **[1.5 poen]** Napiši rekurzivnu funkciju koja računa zbir svih cifara datog broja.
   Primer: n = 12345 → rezultat 15.

```c
int zbir_cifara(int n) {
   if (n == 0) return 0;
   return (n % 10) + zbir_cifara(n / 10);
}
```

9. **[1 poen]** Imamo sledeći kod:

```
#include <stdio.h>

struct Tacka {
   int x;
   int y;
};

int main() {
   struct Tacka niz[4] = {{1,2}, {3,4}, {5,6}, {7,8}};
   struct Tacka *p = niz;

   // pitanje: kako možemo pristupiti elementu 3 i njegovom x članu?
   return 0;
}
```

Pitanje: Koji od sledećih izraza ispravno pristupa x članu trećeg elementa (broju 5)?
Zaokruži sve tačne.
a) niz[2].x
b) p[2].x
c) (p+2)->x
d) *(p+2).x
e) (*(p+2)).x

10. **[1.5 poen]** Napiši funkciju koja vraća broj čvorova u listi.

```
int countNodes(Node *head) {
   int count = 0;
   while (head != NULL) {
      count++;
      head = head->next;
   }
   return count;
}
```

11. **[2 poena]** Kalkulator preko argumenata
Program prima tri argumenta: prvi broj, operator (+, -, *, /), drugi broj.
Izračunati i ispisati rezultat.
Primer:
./a.out 5 + 7
Rezultat: 12

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

int main(int argc, char *argv[]) {
    if (argc != 4) {
        printf("Upotreba: %s broj1 operator broj2\n", argv[0]);
        return 1;
    }

    double a = atof(argv[1]);
    double b = atof(argv[3]);
    char op = argv[2][0];
    double rez;

    switch (op) {
        case '+': rez = a + b; break;
        case '-': rez = a - b; break;
        case '*': rez = a * b; break;
        case '/':
            if (b == 0) {
                printf("Greska: deljenje nulom!\n");
                return 1;
            }
            rez = a / b;
            break;
        default:
            printf("Nepoznat operator!\n");
            return 1;
    }

    printf("Rezultat: %.2f\n", rez);
    return 0;
}
```

12. **[2 poena]** Napiši funkciju koja obrće listu po blokovima od k čvorova.
    Primer: lista 1→2→3→4→5→6, k=2 daje izlaz 2→1→4→3→6→5

```
Node* reverseK(Node* head, int k) {
   Node* prev = NULL;
   Node* curr = head;
   Node* next = NULL;
   int count = 0;

   // obrni prvih k čvorova
   while (curr != NULL && count < k) {
      next = curr->next;
      curr->next = prev;
      prev = curr;
      curr = next;
      count++;
   }

   // curr sada pokazuje na (k+1)-ti čvor
   if (next != NULL) {
      head->next = reverseK(next, k);
   }

   // prev je nova glava bloka
   return prev;
}
```

13. **[2 poena]** Napiši funkciju koja vraća novu listu koja predstavlja presek dve liste.

```
Node* intersection(Node *list1, Node *list2) {
   Node *result = NULL;
   Node *p1 = list1;

   while (p1 != NULL) {
      Node *p2 = list2;
      while (p2 != NULL) {
         if (p1->data == p2->data) {
            // da izbegnemo duplikate u rezultatu
            Node *check = result;
            int exists = 0;
            while (check != NULL) {
               if (check->data == p1->data) {
                  exists = 1;
                  break;
               }
               check = check->next;
            }
            if (!exists) {
               append(&result, p1->data);
            }
```

```
            break; // nema potrebe dalje po drugoj listi
          }
          p2 = p2->next;
        }
        p1 = p1->next;
      }

      return result;
}
```

14. **[2 poena]** Sortiraj niz brojeva 55 12 63 27 25 14 79 36 88 44 koristeći Stack sort algoritam i ispisati svaki korak prilikom sortiranja brojeva.

M 55
V

M 12
V 55

M 12 55 63
V

M 12 27
V 63 55

M 12 25
V 63 55 27

M 12 14
V 63 55 27 25

M 12 14 25 27 55 63 79
V

M 12 14 25 27 36
V 79 63 55

M 12 14 25 27 36 55 63 79 88
V

M 12 14 25 27 36 44
V 88 79 63 55

15. **[2.5 poena]** Napiši program koji:
    - Traži od korisnika broj redova n.
    - Dinamički alocira donju trougaonu matricu tako da red i ima i+1 elemenata (redovi rastu).
    - Popunjava matricu brojevima 1, 2, 3, ... po redu (sledeći broj ide u sledeći element, red po red).
    - Ispisuje matricu tako da svaki red bude u jednoj liniji.
    - Oslobađa svu memoriju.

Ulaz: 4
Izlaz:
1
2 3
4 5 6
7 8 9 10

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

int main() {
    int n, i, j, broj = 1;
    int **mat;

    printf("Unesite dimenziju matrice: ");
    scanf("%d", &n);

    // alokacija pokazivača na redove
    mat = (int **)malloc(n * sizeof(int *));
    if (!mat) {
        printf("Neuspesna alokacija memorije.\n");
        return 1;
    }

    // za svaki red alociramo samo onoliko kolona koliko treba (trougaona)
    for (i = 0; i < n; i++) {
        mat[i] = (int *)malloc((i + 1) * sizeof(int));
        if (!mat[i]) {
            printf("Neuspesna alokacija memorije.\n");
            return 1;
        }
    }

    // popunjavanje trougaone matrice redom brojevima
    for (i = 0; i < n; i++) {
        for (j = 0; j <= i; j++) {
            mat[i][j] = broj++;
        }
    }

    // ispis
    printf("Trougaona matrica:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j <= i; j++) {
            printf("%4d", mat[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```