

1. Što je neispravno u sljedećem programu?

```
#include <stdio.h>
int *vratiDvostuko(int n) {
    int rez;
    rez = 2 * n;
    return &rez;
}

int main(void) {
    int a = 10, *rez = NULL;
    rez = vratiDvostuko(a);
    printf("Rezultat je %d", *rez);
    return 0;
}
```

funkcija nije tipa *pokazivač
dangling pointer

2. Što je neispravno u sljedećem programu?

```
#include <stdio.h>
int main(void) {
    int a = 10, *p1 = &a, b = 20, *p2, c;
    c = *p1 + *p2;
    printf("c = %d", c);
    return 0;
}
```

nije rečeno što je *p2

3. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>
int main(void) {
    int polje[] = {2, 3, 5, 7, 11, 23, 29};
    int *p1 = &polje[0], *p2 = NULL;
    p2 = &polje[5];
    printf ("%d %d\n", *p1, *p2);
    printf ("%d %d\n", *(p2 - 1), *p2 - 1);
    printf ("%d %d\n", p2 - p1, *p2 - *p1);
    return 0;
}
```

udaljenost
pokazivača

*p1 = 2
*p2 = 23
2 23
11 22
24 21
5

4. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

int main(void) {
    int polje[4][3] = {{2, 11, 23},
                      {29, 31, 37},
                      {47, 51, 59},
                      {61, 67, 71}};

    int *p1 = NULL, *p2 = NULL, *p3 = NULL;
    p1 = &polje[0][1]; // 11
    p2 = p1 + 5; // 47
    p3 = &polje[3][1]; // 67
    printf ("%d %d\n", *p1, *p2); // 11 47
    printf ("%d %d\n", *(p1 + 3), *(p1 + 3)); // 31 14
    printf ("%d %d\n", p3 - p2, *p3 - *p2); // 4 20
    return 0;
}
```

5. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

int main(void) {
    int polje[2][3] = {{2, 11, 23}, {47, 51, 59}};
    int *p = &polje[0][1], i = 3;
    p += i; // 51
    *(--p) = --i; // na mjesto 47 → 2
    printf ("%d %d %d %d\n", *p, polje[1][0], polje[1][1], polje[1][2]);
    return 0;
}
```

Handwritten notes and results:

- Initial array: 2 11 23 / 47 51 59
- After `p += i`: `p` points to 51.
- After `*(--p) = --i`: `i` becomes 2, and the value 47 is replaced by 2.
- Final array: 2 11 23 / 2 51 59
- Output: 2 2 51 59

6. Što će se ispisati izvršavanjem sljedećeg programa? Riješiti "na papiru" i rezultat provjeriti izvršavanjem na računalu.

```
#include <stdio.h>

int main(void) {
    int polje[4][3] = {{2, 11, 23},
                      {29, 31, 37},
                      {47, 51, 59},
                      {61, 67, 71}};

    int *p = polje[0]; 2 ✓
    int i1 = (*p)++; prvo i1=2, a onda *p postaje 3
    int i2 = *p++; → i2 je *p pa ne povećava na 11
    printf ("%d %d\n", i1, i2);
                    2 3

    p = &polje[0][0]; 2 → pozicija od [0][0]
    i1 = ++*p; → *p = 3+1=4 ←
    i2 = *++p; 11
    printf ("%d %d\n", i1, i2);
                    4 11

    p = &polje[1][1]; → p=31
    i1 = ++*--p; prvo povećam i2 a onda (29) dodam 1 = 30
    p = &polje[2][2]; 59
    i2 = ++*p--; 59+1 = 60 i to povećam na mjesto od 60
    printf ("%d %d\n", i1, i2);
                    30 60

    return 0;
}
```

7. Što će se ispisati izvršavanjem sljedećeg programa?

```

#include <stdio.h>

void f(int *p, int n) {
    printf ("%d\n", *(p + n));
}

int main(void) {
    int polje[] = {2, 11, 23, 29, 31, 37};
    int *pp;
    pp = &polje[0];
    f(pp++, 2); (2, 2)
    f(pp, 2); (3, 2)
    f(++pp, 2); (4, 2)

    return 0;
}

```

$p[0] \rightarrow 23$
 $p[1] \rightarrow 29$
 31

8. Što će se ispisati izvršavanjem sljedećeg programa?

```

#include <stdio.h>

int *f(int *p) {
    return p - *p;
}

int main(void) {
    int polje[5][3] = {{1, -1, 3},
                      {-3, 2, -2},
                      {0, 5, -4},
                      {-4, -6, 2},
                      {2, 1, 0}};

    int *p = &polje[1][1]; 2
    p = f(p); p[0][2]
    printf ("%d\n", *p); 3

    p = f(p + *p); (-2)
    printf ("%d\n", *p); 5

    return 0;
}

```

$[0, 2] + 3$
 $f(p + *p) \rightarrow -2$
 $\text{return}([1][2] - (-2))$
 $\rightarrow \text{pomaćemo}$
 $\text{unaprijed za } 2$
 $\rightarrow 5$

9. Napisati funkciju `zbroy2D` koja kao rezultat vraća zbroj svih članova matrice (članovi matrice su tipa `double`) zadanih dimenzija $m \times n$. Napisati glavni program (funkciju `main`) tako da s tipkovnice učitava dimenzije i članove matrice te na zaslon ispiše rezultat dobiven pozivom funkcije `zbroy2D`, sukladno prikazanom primjeru.

Primjer izvršavanja programa.

```
Upisite dimenzije > 3 4↵
Upisite članove >↵
1.2 -4.1 2.2 8.15↵
415.9 1.0 1.0 1.0↵
2.2 2.2 2.2 12.2↵
Suma je: 445.150000
```

10. Napisati funkciju `transpKvad` koja transponira zadanu kvadratnu matricu (pri tome samo mijenja članove zadane matrice, dakle ne stvara novu matricu).

Napisati glavni program (funkciju `main`) tako da s tipkovnice učitava red matrice i članove. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `transpKvad` sukladno prikazanom primjeru (za ispis svakog pojedinog člana matrice koristi se format `"%5d"`).

Primjer izvršavanja programa.

```
Upisite red matrice > 4↵
Upisite članove >↵
10 20 30 40↵
11 21 31 41↵
12 22 32 42↵
13 23 33 43↵
 10  11  12  13↵
 20  21  22  23↵
 30  31  32  33↵
 40  41  42  43↵
```

11. Napisati funkciju `genPrim` koja za zadane parametre granica i n vraća n prim brojeva koji su veći ili jednaki parametru granica.

Napisati glavni program (funkciju `main`) tako da s tipkovnice učitava vrijednost donje granice i broj prim brojeva koje treba ispisati. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `genPrim` sukladno prikazanom primjeru (za ispis svakog pojedinog prim broja koristi se format `"%7d"`).

Primjer izvršavanja programa.

```
Upisite donju granicu i broj prim brojeva > 5000 4↵
 5003↵
 5009↵
 5011↵
 5021↵
```

12. Napisati funkciju `sort1D` koja sortira zadano jednodimenzijско polje cijelih brojeva. Funkcija također kao parametar prima logičku vrijednost `silazno`. Ako je `silazno` istina, tada članove polja treba sortirati od većih prema manjim vrijednostima, inače, treba ih sortirati uzlazno, od manjih prema većim. Napisati glavni program (funkciju `main`) tako da s tipkovnice učitava jedan znak (ako je učitani znak 'S', sortiranje će trebati obaviti silazno, inače uzlazno), dimenziju i članove polja. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `sort1D` (dakle sadržaj poredanog polja), sukladno prikazanom primjeru (za ispis svakog pojedinog člana polja koristi se format `"%d·"`).

Primjeri izvršavanja programa.

```
Upisite smjer poretka (S-silazno) > S
Upisite dimenziju > 7
Upisite članove > 7 3 12 4 4 3 5
12·7·5·4·4·3·3·
```

```
Upisite smjer poretka (S-silazno) > s
Upisite dimenziju > 7
Upisite članove > 7 3 12 4 4 3 5
3·3·4·4·5·7·12·
```

13. Napisati funkciju `sortRetke2D` koja unutar svakog retka sortira članove zadanog dvodimenzijскоg polja cijelih brojeva. Funkcija također kao parametar prima logičku vrijednost `silazno`. Ako je `silazno` istina, tada članove polja unutar svakog retka treba sortirati od većih prema manjim vrijednostima, inače, treba ih sortirati uzlazno, od manjih prema većim. Za sortiranje članova retka treba koristiti funkciju `sort1D` iz prethodnog zadatka.

Napisati glavni program (funkciju `main`) tako da s tipkovnice učitava jedan znak (ako je učitani znak 'S', sortiranje će trebati obaviti silazno, inače uzlazno), zatim učitava dimenzije i članove polja. Nakon toga na zaslon treba ispisati rezultat poziva funkcije `sortRetke2D` (dakle sadržaj polja kojem su članovi unutar redaka sortirani) sukladno prikazanom primjeru (za ispis svakog pojedinog člana polja koristi se format `"%5d"`).

Primjeri izvršavanja programa.

```
Upisite smjer poretka (S-silazno) > S
Upisite dimenzije > 2 4
Upisite članove >
1 2 3 2
5 4 1 2
3 2 2 1
5 4 2 1
```

```
Upisite smjer poretka (S-silazno) > s
Upisite dimenzije > 2 4
Upisite članove >
1 2 3 2
5 4 1 2
1 2 2 3
1 2 4 5
```

shpaac
for (i=0; i<n; i++)
for (j=i+1; j<n; j++)
if (*(p+j) > *(p+i))
pau = p+i
*(p+i) = *(p+j)
*(p+j) = pau

tako za jedan red
red
možemo
2 funkcije
 $p + i * m + j$
 $p + k * m + i$
treba staviti N

1232
5412

2 > 1 → 2232
2132 ✓

14. Napisati funkciju `negativci` koja za zadano jednodimenzijско polje vraća sve negativne članove tog polja.

Napisati glavni program (funkciju `main`) tako da s tipkovnice učitа dimenziju i članove polja. Nakon toga pomoću rezultata poziva funkcije `negativci` ispisati negativne članove polja sukladno prikazanom primjeru (za ispis svakog pojedinog člana polja koristi se format `"%d · "`).

Primjeri izvršavanja programa.

```
Upisite broj članova > 10↵
Upisite članove > 1 2 3 2 -1 6 -4 -1 -2 3↵
-1·-4·-1·-2·
```

```
Upisite broj članova > 10↵
Upisite članove > 1 2 3 2 1 6 4 1 2 3↵
```

15. Napisati funkciju `prviNegativac` koja za zadano jednodimenzijско polje vraća pokazivač na prvi pronađeni negativni član u polju.

Napisati glavni program (funkciju `main`) tako da s tipkovnice učitа dimenziju i članove polja. Nakon toga ispisati rezultat poziva funkcije `prviNegativac` sukladno prikazanom primjeru.

Primjeri izvršavanja programa.

```
Upisite broj članova > 10↵
Upisite članove > 1 2 3 2 -1 6 -4 -1 -2 3↵
Prvi negativni je -1
```

```
Upisite broj članova > 10↵
Upisite članove > 1 2 3 2 1 6 4 1 2 3↵
Nema negativnih
```

Rješenja:

1. Funkcija vraća pokazivač na objekt (varijablu) koji je definiran u funkciji. Taj objekt više ne postoji kada funkcija završi (funkcija je vratila viseći pokazivač - *dangling pointer*).
2. Varijabla p2 nije inicijalizirana, stoga sadrži pokazivač koji pokazuje na neodređeno mjesto u memoriji. Rezultat izvršavanja programa je nepredvidiv.

3. -

4. -

5. -

6. -

7. -

8. -

9. `#include <stdio.h>`

```
double zbroj2D(double *mat, int m, int n) {
    int i;
    double suma = 0.;
    // moglo je s dvije petlje, ali nije potrebno jer su svi članovi u memoriji jedan iza drugog
    for (i = 0; i < m * n; ++i) {
        suma += *(mat + i);
    }

    return suma;
}
```

```
int main(void) {
    int m, n, i, j;
    printf("Upisite dimenzije > ");
    scanf("%d %d", &m, &n);

    double mat[m][n];

    printf("Upisite članove >\n");
    for (i = 0; i < m; ++i) {
        for (j = 0; j < n; ++j) {
            scanf("%lf", &mat[i][j]);
        }
    }

    printf("Suma je: %lf", zbroj2D(&mat[0][0], m, n));
}
```



```
    return 0;
}
```

10. #include <stdio.h>

```
void transpKvad(int *mat, int n) {
    int i, j, pomocna;
    for (i = 0; i < n - 1; ++i) {
        for (j = i + 1; j < n; ++j) {
            pomocna = *(mat + n * i + j);
            *(mat + n * i + j) = *(mat + n * j + i);
            *(mat + n * j + i) = pomocna;
        }
    }
    return;
}

int main(void) {
    int n, i, j;
    printf("Upisite red matrice > ");
    scanf("%d", &n);

    int mat[n][n];

    printf("Upisite clanove >\n");
    for (i = 0; i < n; ++i) {
        for (j = 0; j < n; ++j) {
            scanf("%d", &mat[i][j]);
        }
    }

    transpKvad(&mat[0][0], n);

    for (i = 0; i < n; ++i) {
        for (j = 0; j < n; ++j) {
            printf("%5d", mat[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

```
11. #include <stdio.h>
#include <stdbool.h>
#include <math.h>

void genPrim(int granica, int n, int *rez) {
    bool djeljiv;
    int nadjenoBrojeva = 0, kandidat = granica > 1 ? granica : 2, i;
    while (nadjenoBrojeva < n) {
        i = 2;
        djeljiv = 0;
        while (i <= sqrt(kandidat) && djeljiv == 0) {
            if (kandidat % i == 0) {
                djeljiv = 1;
            }
            i = i + 1;
        }
        if (djeljiv == 0 || kandidat == 1) {
            *(rez + nadjenoBrojeva) = kandidat;
            ++nadjenoBrojeva;
        }
        ++kandidat;
    }
    return;
}

int main(void) {
    int dgr, n, i;
    printf("Upisite donju granicu i broj prim brojeva > ");
    scanf("%d %d", &dgr, &n);

    int primBrojevi[n];

    genPrim(dgr, n, &primBrojevi[0]);

    for (i = 0; i < n; ++i) {
        printf("%7d\n", primBrojevi[i]);
    }

    return 0;
}
```

```
12. #include <stdio.h>
#include <stdbool.h>

void zamijeni(int *x, int *y) {
    int pom;
    pom = *x;
    *x = *y;
    *y = pom;
    return;
}

void sort1D(int *polje, int n, bool silazno) {
    int ind_min_max, i, j;
    for (i = 0; i < n - 1; ++i) {
        ind_min_max = i + 1;
        if (silazno) {
            for (j = i + 2; j < n; ++j) {
                if (*(polje + j) > *(polje + ind_min_max)) ind_min_max = j;
            }
            if (*(polje + ind_min_max) > *(polje + i)) {
                zamijeni(polje + ind_min_max, polje + i);
            }
        } else {
            for (j = i + 2; j < n; ++j) {
                if (*(polje + j) < *(polje + ind_min_max)) ind_min_max = j;
            }
            if (*(polje + ind_min_max) < *(polje + i)) {
                zamijeni(polje + ind_min_max, polje + i);
            }
        }
    }
    return;
}

int main(void) {
    char smjerSorta;
    int n, i;
    printf("Upisite smjer poretka (S-silazno) > ");
    scanf("%c", &smjerSorta);

    printf("Upisite dimenziju > ");
    scanf("%d", &n);

    int polje[n];
    printf("Upisite članove > ");

    for (i = 0; i < n; ++i) {
        scanf("%d", &polje[i]);
    }

    sort1D(polje, n, smjerSorta == 'S');

    for (i = 0; i < n; ++i) {
        printf("%d ", polje[i]);
    }

    return 0;
}
```

```

13. #include <stdio.h>
#include <stdbool.h>

void zamijeni(int *x, int *y) {
    int pom;
    pom = *x;
    *x = *y;
    *y = pom;
    return;
}

void sort1D(int *polje, int n, bool silazno) {
    int ind_min_max, i, j;
    for (i = 0; i < n - 1; ++i) {
        ind_min_max = i + 1;
        if (silazno) {
            for (j = i + 2; j < n; ++j) {
                if (*(polje + j) > *(polje + ind_min_max)) ind_min_max = j;
            }
            if (*(polje + ind_min_max) > *(polje + i)) {
                zamijeni(polje + ind_min_max, polje + i);
            }
        } else {
            for (j = i + 2; j < n; ++j) {
                if (*(polje + j) < *(polje + ind_min_max)) ind_min_max = j;
            }
            if (*(polje + ind_min_max) < *(polje + i)) {
                zamijeni(polje + ind_min_max, polje + i);
            }
        }
    }
    return;
}

void sort2D(int *polje, int m, int n, bool silazno) {
    int i;
    for (i = 0; i < m; ++i) {
        sort1D((polje + n * i + 0), n, silazno);
    }
    return;
}

int main(void) {
    char smjerSorta;
    int m, n, i, j;
    printf("Upisite smjer poretka (S-silazno) > ");
    scanf("%c", &smjerSorta);
    printf("Upisite dimenzije > ");
    scanf("%d %d", &m, &n);

    int mat[m][n];
    printf("Upisite članove >\n");
    for (i = 0; i < m; ++i) {
        for (j = 0; j < n; ++j) {
            scanf("%d", &mat[i][j]);
        }
    }
    sort2D(&mat[0][0], m, n, smjerSorta == 'S');
    for (i = 0; i < m; ++i) {
        for (j = 0; j < n; ++j) {
            printf("%5d", mat[i][j]);
        }
        printf("\n");
    }
    return 0;
}

```

14. #include <stdio.h>

```
int negativci(int *polje, int n, int *nadjeniNegativci) {
    int nNegativaca = 0, i;
    for (i = 0; i < n; ++i) {
        if (*(polje + i) < 0) {
            *(nadjeniNegativci + nNegativaca) = *(polje + i);
            ++nNegativaca;
        }
    }
    return nNegativaca;
}

int main(void) {
    int n, i, nNegativaca;

    printf("Upisite broj clanova > ");
    scanf("%d", &n);

    int polje[n];
    printf("Upisite clanove > ");

    for (i = 0; i < n; ++i) {
        scanf("%d", &polje[i]);
    }

    int nadjeniNegativci[n];
    nNegativaca = negativci(polje, n, nadjeniNegativci);

    for (i = 0; i < nNegativaca; ++i) {
        printf("%d ", nadjeniNegativci[i]);
    }

    return 0;
}
```

15. #include <stdio.h>

```
int *prviNegativac(int *polje, int n) {
    int i;
    for (i = 0; i < n; ++i) {
        if (*(polje + i) < 0) {
            return polje + i;
        }
    }
    return NULL;
}

int main(void) {
    int n, i, *pokNaNegativca;

    printf("Upisite broj clanova > ");
    scanf("%d", &n);

    int polje[n];
    printf("Upisite clanove > ");

    for (i = 0; i < n; ++i) {
        scanf("%d", &polje[i]);
    }

    pokNaNegativca = prviNegativac(polje, n);
    if (pokNaNegativca == NULL) {
        printf("Nema negativnih");
    } else {
        printf("Prvi negativni je %d", *pokNaNegativca);
    }

    return 0;
}
```