OLIMPIADA KOMBËTARE E INFORMATIKËS
NË ARSIMIN E MESËM TË LARTË

Faza e tretë

Udhëzime për nxënësin: **Viti shkollor 2023 - 2024**

- Olimpiada fillon në orën 10:00 dhe mbaron në orën 13:00.
- Testi përmban 5 pyetje.
- Për të zgjidhur secilin ushtrim nxënësi mund të përdorë gjuhën e programimit: C ose C++
- Zgjidhjet do të bëhen në kompjuter.

Për përdorim nga komisioni i vlerësimit

Pyetja	**1**	**2**	**3**	**4**	**5**
	9 pikë	**15 pikë**	**8 pikë**	**11 pikë**	**7 pikë**
Pikët e fituara					

Totali i pikëve të fituara ☐

KOMISIONI I VLERËSIMIT

1.....

2. .....

Ushtrim 1. Shkruani një program i cili shfaq të gjitha kombinimet e mundshme duke mbledhur, zbritur numrat 1,2,3,4,5,6,7,8,9 ose bashkimet e tyre, në mënyrë që rezultati të jetë 100. Renditja e numrave nuk duhet të ndryshojë.

Shembull: 1 + 2 + 34 - 5 + 67 - 8 + 9 **9 pikë**
123 - 4 - 5 - 6 - 7 + 8 – 9

Zgjidhja e ushtrimit 1:

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

// Funksioni për të gjeneruar dhe kontrolluar në mënyrë rekursive të gjitha mundësitë
void gjeneroShprehjet(int numriAktual, int shumaDeriTani, int numriPararendës, char *shprehja) {
   // Rasti bazë: nëse janë përdorur të gjithë numrat
   if (numriAktual == 10) {
      // Nëse shuma është 100, printo shprehjen
      if (shumaDeriTani == 100) {
         printf("%s\n", shprehja);
      }
      return;
   }
   // Konverto numrin aktual në string
   char numriStr[2];
   sprintf(numriStr, "%d", numriAktual);

   // Shto numrin aktual në shprehje
   char shprehjaPlus[50], shprehjaMinus[50], shprehjaConcat[50];
   sprintf(shprehjaPlus, "%s + %s", shprehja, numriStr);
   sprintf(shprehjaMinus, "%s - %s", shprehja, numriStr);

   // Thirrje rekursive me mbledhje
   gjeneroShprehjet(numriAktual + 1, shumaDeriTani + numriAktual, numriAktual, shprehjaPlus);
   // Thirrje rekursive me zbritje
   gjeneroShprehjet(numriAktual + 1, shumaDeriTani - numriAktual, -numriAktual, shprehjaMinus);
```

```
    // Trajto Concat në mënyrën e duhur
    sprintf(shprehjaConcat, "%s%s", shprehja, numriStr);
    gjeneroShprehjet(numriAktual + 1, shumaDeriTani - numriPararendës + (numriPararendës * 10 + (numriPararendës> 0 ? numriAktual : -numriAktual)), numriPararendës * 10 + (numriPararendës > 0 ? numriAktual : -numriAktual), shprehjaConcat);
}
int main() {
    // Filloni me numrin e parë
    gjeneroShprehjet(2, 1, 1, "1");
    return 0;
}
```

Ushtrim 2. **15 pikë**

Ka dy lloje mjetesh transporti në Tiranë:
Autobus i cili ka një kapacitet prej 100 personash.
Makinë e cila ka një kapacitet prej 4 personash.

Ka N njerëz që duan të udhëtojnë nga vendi A në vendin B. Një autobus emeton X njësi CO_2 ndërsa një makinë emeton Y njësi CO_2 në udhëtimin e tyre nga A në B.

Shkruani një program i cili organizon autobusët dhe makinat për të transportuar njerëzit në mënyrë që të minimizohet sasia e CO_2 e emetuar nga mjetet e transportit. Llogarisni vlerën optimale të sasisë së CO_2 të emetuar.

Input:

Përdoruesi jep tre numra të plotë N, X, Y - përkatësisht numrin e njerëzve që duan të udhëtojnë, njësitë e CO_2 të lëshuar nga një autobus dhe njësitë e CO_2 të lëshuar nga një makinë.

Output:

Llogaritet kombinimi i numrit të autobusëve dhe numrit të makinave i cili rezulton në sasinë minimale të emetimit të CO_2 dhe vlera e sasisë së CO_2 të emetuar.

Shembull:
Input: 1200, 250, 5
Output: NrA = 0 autobusë, NrM= 300 makina, Sasia = 1500 njësi CO_2

Zgjidhja e ushtrimit 2:

```
#include <stdio.h>
int main()
{
    // Vendos të dhënat
    int NrAutobusë = 0;
```

```
int NrMakina = 0;
int EficensëAutobus = 0;
int EficensëMakinë = 0;
int n = 0;
printf("Vendosi numrin e udhëtarëve \n");
scanf("%d",&n);
 printf("Vendosni njësi CO2 të lëshuar nga autobusi \n");
scanf("%d",  & EficensëAutobus);
 printf("Vendosni njësi CO2 të lëshuar nga makina \n");
scanf("%d",& EficensëMakinë);
if (EficensëAutobus < (EficensëMakinë * 25)) // Autobuzat janë më optimalë
{
   while (n >= 100) // Vendos njerëzit në autobusë
   {
      NrAutobusë++;
      n -= 100;
   }
   // Kontrollojmë nëse njerëzit e mbetur është më optimale të vendosen në autobusë
   if (n % 4 == 0)
   {
      if ((n / 4 * EficensëMakinë) > EficensëAutobus) // Kontrollojmë nëse është më optimale të vendosen në një autobus apo disa makina
      {
         NrAutobusë ++;
         n = 0;
      }
      else // Më optimale është të përdoren makina
      {
         while (n > 0)
         {
            NrMakina ++;
             n -= 4;
         }
      }
   }
   else
   {
      if ((n / 4 * EficensëMakinë + 1) > EficensëAutobus) // Kontrollojmë nëse është më optimale të vendosen në autobus apo në makina
      {
         NrAutobusë ++;
         n = 0;
      }
      else // Më optimale është të përdoren makina
      {
         while (n > 0)
         {
            NrMakina ++;
            n -= 4;
         }
```

```
        }
      }
   }
   else
   {
      // Njerëzit e mbetur i vendosim në makina
      while (n > 0)
      {
         NrMakina ++;
         n -= 4;
      }
   }
   printf("NrAutobusë = %d autobusë, NrMakina = %d makina, Sasia optimale e emetuar = %d njësi CO2.", NrAutobusë, NrMakina, NrAutobusë * EficensëAutobus + NrMakina * EficensëMakinë);
   return 0;
}
```

Ushtrim 3. **8 pikë**

Konsideroni një kopsht të përfaqësuar si një rrjet n x m, ku çdo qelizë njësi ka përmasa 1 x 1 dhe përmban një vazo lulesh. Për të siguruar kushtet optimale të rritjes së bimëve, vazot duhet të kenë hapësirë ndërmjet njëra tjetrës. Shkruani një program që përcakton numrin maksimal dhe minimal të bimëve që mund të mbillen në mënyrë strategjike në kopsht, duke iu përmbajtur kufizimit që nuk mund të mbillen dy bimë radhazi në të njëjtin rresht.

Zgjidhja e ushtrimit 3:

```
#include <stdio.h>
#include <math.h>
// Funksioni për gjetjen e numrit maksimal të bimëve
int maksBimë(int n, int m) {
   return ceil(m / 2.0) * n;
}
// Funksioni për gjetjen e numrit minimal të bimëve
int minBimë(int n, int m) {
   if (m == 1) {
      return ceil(m / 2.0) * n;
   } else {
      return floor(m / 2.0) * n;
   }
}
int main() {
```

```
    int n, m;
    printf("Vendos numrin e rreshtave dhe kolonave në kopsht: ");
    scanf("%d %d", &n, &m);

    int maksBimës = maksBimë(n, m);
    int minBimës = minBimë(n, m);
    printf("Numri maksimal i bimëve që mund të mbillen: %d\n", maksBimës);
    printf("Numri minimal i bimëve të nevojshëm: %d\n", minBimës);
    return 0;
}
```

Ushtrim 4.

Jepet një tabelë shahu dhe një gur kalorësi i vendosur fillimisht në pozicionin e pasqyruar në figurën 1. Shkruani një program i cilin gjen një sekuencë lëvizjesh të tilla që kalorësi të vizitojë çdo katror të shahut saktësisht një herë. Lëvizja e kalorësit është në formën e shkronjës L. Në figurën 2 pasqyrohen të gjitha lëvizjet e mundshme të kalorësit nga një pozicion i caktuar. **11 pikë**

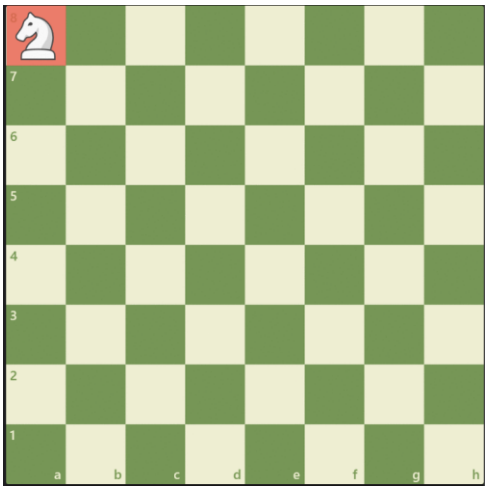

Figura 1

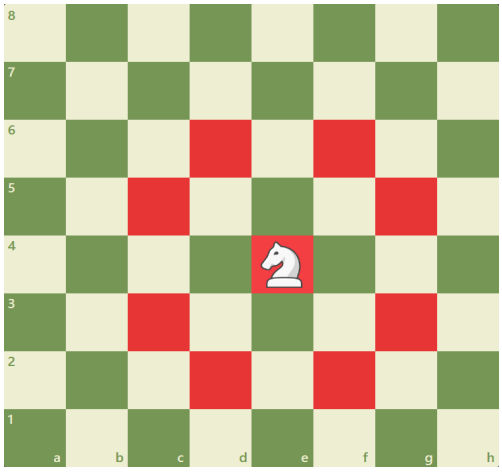

Figura 2

Input: N=8

Output:

0	59	38	33	30	17	8	63
37	34	31	60	9	62	29	16
58	1	36	39	32	27	18	7
35	48	41	26	61	10	15	28
42	57	2	49	40	23	6	19
47	50	45	54	25	20	11	14
56	43	52	3	22	13	24	5
51	46	55	44	53	4	21	12

Zgjidhja e ushtrimit 4:

```
#include <stdio.h>
#define N 8

// Funksioni që verifikon nëse një lëvizje është e sigurt
int ështëEsigurt(int x, int y, int sol[N][N]) {
   return (x >= 0 && x < N && y >= 0 && y < N && sol[x][y] == -1);
```

```
}

// Funksioni për të printuar zgjidhjen
void printoZgjidhjen(int sol[N][N]) {
   for (int x = 0; x < N; x++) {
      for (int y = 0; y < N; y++)
         printf(" %2d ", sol[x][y]);
      printf("\n");
   }
}

// Funksioni për të zgjidhur problemin e Kalorësit në mënyrë rekursive
int zgjidhjeKalorësiUtil(int x, int y, int lëvizje, int sol[N][N], int xLëvizje[], int yLëvizje[]) {
   int k, next_x, next_y;
   if (lëvizje == N * N)
      return 1;

   for (k = 0; k < 8; k++) {
      next_x = x + xLëvizje[k];
      next_y = y + yLëvizje[k];
      if (ështëEsigurt(next_x, next_y, sol)) {
         sol[next_x][next_y] = lëvizje;
         if (zgjidhjeKalorësiUtil(next_x, next_y, lëvizje + 1, sol, xLëvizje, yLëvizje))
            return 1;
         else
            sol[next_x][next_y] = -1;
      }
   }
   return 0;
}

// Funksioni për të zgjidhur problemin e Kalorësit
int zgjidhjeKalorësi() {
   int sol[N][N];
```

```
    for (int x = 0; x < N; x++)
        for (int y = 0; y < N; y++)
            sol[x][y] = -1;

    int xLëvizje[8] = {2, 1, -1, -2, -2, -1, 1, 2};
    int yLëvizje[8] = {1, 2, 2, 1, -1, -2, -2, -1};

    sol[0][0] = 0;

    if (zgjidhjeKalorësiUtil(0, 0, 1, sol, xLëvizje, yLëvizje) == 0) {
        printf("Zgjidhja nuk ekziston");
        return 0;
    } else
        printoZgjidhjen(sol);

    return 1;
}

// Funksioni kryesor
int main() {
    zgjidhjeKalorësi();
    return 0;
}
```

Ushtrimi 5. **7 pikë**

Një platformë e rrjeteve sociale përmban informacionin mbi përdoruesit e platformës dhe aktivitetet e tyre. Çdo përdorues ka një numër identifikues unik dhe një set të dhënash, i cili përfshin: numrin e postimeve të bëra, numrin e ndjekësve, numrin e pëlqimeve dhe numrin e komenteve për çdo përdorues.

Shkruani një program i cili:

Përcakton 5 përdoruesit e platformës me numrin më të ulët të ndjekësve.

Përcakton 3 përdoruesit më aktivë në platformë. Këta përdorues kanë numrin më të madh të pëlqimeve dhe komenteve të kombinuara së bashku.

Përcakton shkallën mesatare të aktivitetit të platformës. Aktiviteti i platformës matet duke llogaritur numrin e pëlqimeve dhe komenteve për postim.

Zgjidhja e ushtrimit 5

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>  // Për INT_MAX

#define NUM_USERS 10

// Struktura për të përfaqësuar informacionin e një përdoruesi
typedef struct {
    int user_id;
    int num_posts;
    int num_followers;
    int num_likes;
    int num_comments;
} User;

// Funksioni për gjetjen e 5 përdoruesit me numrin më të ulët të ndjekësve
void findUsersWithLowestFollowers(User users[], int n) {
    printf("5 përdoruesit me numrin më të ulët të ndjekësve:\n");
    int lowest_followers_ids[5] = {0};
    int lowest_followers[5] = {INT_MAX};  // Përdorja e INT_MAX

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < 5; j++) {
            if (users[i].num_followers < lowest_followers[j]) {
                for (int k = 4; k > j; k--) {
                    lowest_followers[k] = lowest_followers[k - 1];
                    lowest_followers_ids[k] = lowest_followers_ids[k - 1];
                }
                lowest_followers[j] = users[i].num_followers;
                lowest_followers_ids[j] = users[i].user_id;
                break;
```

```
            }
        }
    }

    for (int i = 0; i < 5; i++) {
        printf("Përdoruesi %d: %d ndjekës\n", lowest_followers_ids[i], lowest_followers[i]);
    }
    printf("\n");
}

// Funksioni për gjetjen e 3 përdoruesit më aktivë
void findMostActiveUsers(User users[], int n) {
    printf("3 përdoruesit më aktivë:\n");
    int most_active_user_ids[3] = {0};
    int most_activity[3] = {0};

    for (int i = 0; i < n; i++) {
        int combined_activity = users[i].num_likes + users[i].num_comments;
        for (int j = 0; j < 3; j++) {
            if (combined_activity > most_activity[j]) {
                for (int k = 2; k > j; k--) {
                    most_activity[k] = most_activity[k - 1];
                    most_active_user_ids[k] = most_active_user_ids[k - 1];
                }
                most_activity[j] = combined_activity;
                most_active_user_ids[j] = users[i].user_id;
                break;
            }
        }
    }

    for (int i = 0; i < 3; i++) {
        printf("Përdoruesi %d: %d pëlqime dhe %d komente\n", most_active_user_ids[i], users[most_active_user_ids[i] -
1].num_likes, users[most_active_user_ids[i] - 1].num_comments);
    }
    printf("\n");
}

// Funksioni për gjetjen e shkallës mesatare të aktivitetit të platformës
void calculateAverageActivity(User users[], int n) {
    int total_likes = 0, total_comments = 0;

    for (int i = 0; i < n; i++) {
        total_likes += users[i].num_likes;
        total_comments += users[i].num_comments;
    }

    double average_likes = (double)total_likes / n;
    double average_comments = (double)total_comments / n;
```

```
    printf("Shkalla mesatare e aktivitetit:\n");
    printf("Mesatarja e pëlqimeve për postim: %.2lf\n", average_likes);
    printf("Mesatarja e komenteve për postim: %.2lf\n", average_comments);
}

int main() {
    // Array për të përfaqësuar përdoruesit
    User users[NUM_USERS] = {
        {1, 5, 100, 20, 30},
        {2, 8, 80, 15, 25},
        {3, 6, 150, 10, 40},
        {4, 3, 70, 5, 15},
        {5, 10, 200, 25, 50},
        {6, 4, 90, 8, 20},
        {7, 9, 180, 18, 35},
        {8, 7, 120, 12, 30},
        {9, 2, 60, 4, 10},
        {10, 12, 250, 30, 60}
    };

    // Gjejmë 5 përdoruesit me numrin më të ulët të ndjekësve
    findUsersWithLowestFollowers(users, NUM_USERS);

    // Gjejmë 3 përdoruesit më aktivë
    findMostActiveUsers(users, NUM_USERS);

    // Llogarisim shkallën mesatare të aktivitetit të platformës
    calculateAverageActivity(users, NUM_USERS);

    return 0;
}
```

Shënim Kërkesa e tretë e Ushtrimit të pestë konsiderohet e zgjidhur saktë edhe nëse shkalla mesatare e aktivitetit të platformës është llogaritur si mesatarja e pëlqimeve dhe komenteve të kombinuara së bashku.*