17. veljače 2023.

2023 Natjecanje iz informatike

Županijska razina 2023./ Osnovna škola (5. i 6. i 7. i 8. razred)

Primjena algoritama OŠ

OPISI ALGORITAMA

Agencija za odgoj i obrazovanje
Education and Teacher Training Agency

HRVATSKI SAVEZ INFORMATIČARA

Ministarstvo znanosti, obrazovanja i sporta

5.1. Zadatak: Brojiš Autor: Nikola Dmitrović

Prolazimo brojeve od A do B te ako je trenutni broj djeljiv s 10, ispišemo ga, a ako nije, ispišemo znamenku jedinica tog broja.

Programski kod (pisan u Python 3)

```
A = int(input())
B = int(input())

for i in range(A, B + 1):
    if i % 10 == 0:
        print(i)
    else:
        print(i % 10)
```

Potrebno znanje: naredba ponavljanja, dijeljenje s ostatkom

Kategorija: ad hoc

5.2. Zadatak: Bodovi Autor: Nikola Dmitrović

Provjerom svim mogućnosti određujemo sve vrijednosti bodova koje je bilo moguće dobiti na testu. Usput pratimo je li X jednak nekom od tih brojeva te na kraju odredimo ispis „DA“/“NE“.

Programski kod (pisan u Python 3)

```
B1 = int(input())
B2 = int(input())
B3 = int(input())
X = int(input())

nasli = 0
print(0, end = ' ')
if X == 0: nasli = 1

trenutno = B1
print(B1, end = ' ')
if X == B1: nasli = 1

if B2 != trenutno:
    print(B2, end = ' ')
    trenutno = B2
    if X == B2: nasli = 1
```

```
if B3 != trenutno:
    print(B3, end = ' ')
    trenutno = B3
    if X == B3: nasli = 1

if B1 + B2 != trenutno:
    print(B1 + B2, end = ' ')
    trenutno = B1 + B2
    if X == B1 + B2: nasli = 1

if B1 + B3 != trenutno:
    print(B1 + B3, end = ' ')
    trenutno = B1 + B3
    if X == B1 + B3: nasli = 1

if B2 + B3 != trenutno:
    print(B2 + B3, end = ' ')
    trenutno = B2 + B3
    if X == B2 + B3: nasli = 1

if B1 + B2 + B3 != trenutno:
    print(B1 + B2 + B3)
    trenutno = B1 + B2 + B3
    if X == B1 + B2 + B3: nasli = 1
if nasli:
    print("DA")
else:
    print("NE")
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

5.3. Zadatak: Podjela — Autor: Nikola Dmitrović

Simulacijom koraka iz teksta zadatka dolazimo do rješenja.

Programski kod (pisan u Python 3)

```
X = int(input())
```

```
Y = int(input())
Z = int(input())
N = max(X, Y, Z)
A = B = C = 0
for i in range(N):
    if X > 0:
        A += 20
        X -= 1
        if Y > 0:
            B += 10
            Y -= 1
            if Z > 0:
                C += 5
                Z -= 1
        else:
            if Z > 0:
                B += 5
                Z -= 1
    else:
        if Y > 0:
            A += 10
            Y -= 1
            if Z > 0:
                B += 5
                Z -= 1
        else:
            if Z > 0:
                A += 5
                Z -= 1
print(A)
print(B)
print(C)
```

Potrebno znanje: naredba ponavljanja

Kategorija: ad hoc

6.1. Zadatak: Bodovi Autor:

Vidi 5.2

6.2. Zadatak: Prekid Autor: Nikola Dmitrović

Kontinuiranim praćenjem rezultata po minutama, možemo odrediti odgovore na postavljana pitanja. Neovisno o ukupnom rješenju, odgovor na prvo pitanje moguće je dobiti zbrajanjem vrijednosti u prvom retku te u drugom retku ulaza.

Programski kod (pisan u Python 3)

```
N = int (input())

vatreni = list(map(int, input().split()))
protivnici = list(map(int, input().split()))
for i in range(1, N):
    vatreni[i] = vatreni[i-1] + vatreni[i]
for i in range(1, N):
    protivnici[i] = protivnici[i-1] + protivnici[i]
razlika = [0] * N
for i in range(N):
    razlika[i] = vatreni[i] - protivnici[i]

# prvo pitanje
print(vatreni[N-1], protivnici[N-1])
ishod = max(razlika)

if ishod > 0:
    print('W')
    for i in range(N):
        if razlika[i] == ishod:
            gdje = i
elif ishod == 0:
    print('D')
    for i in range(N):
        if razlika[i] == ishod:
            gdje = i
else:
    print('L')
    golova = 0
```

```
    for i in range(N):
        if razlika[i] == ishod and protivnici[i] == 1:
            gdje = i
print(gdje + 1)
```

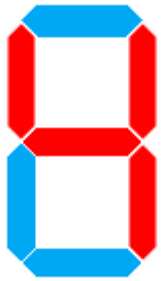
Potrebno znanje: lista, simulacija

Kategorija: ad hoc

6.3. Zadatak: Lift

Autor: Vito Anić

Odredimo kad je broj četiri moguće prikazati na zaslonu. Najjednostavniji način za riješiti taj problem je da proučavamo koje lampice moraju sigurno biti ugašene da bi mogli dobiti traženi broj. Za broj četiri to su lampice A, E i D. (označeno s plavom bojom na slici desno) Ako su sve te tri lampice na trenutnom displeju ugašene, onda možemo dobiti broj četiri s trenutnim zaslonom.

 Prikaz broja 4.	 Na ovom se zaslonu mogao nalaziti broj 4	 Na ovom se zaslonu nije mogao nalaziti broj 4

Tako možemo za svaku znamenku odrediti je li njen prikaz moguć s zaslonom ili nije. Tako smo mogli riješiti problem u Mirkovoj zgradi. Preostaje nam ga riješiti u Slavkovoj. Odredimo za dvije znamenke u Slavkovoj zgradi zasebno koje su mogle znamenke na tim ekranima biti. Na primjer neka su na prvoj mogli biti brojevi 0, 2, 4 i 8, a na drugoj 9 i 8. Tada je ukupno moguće biti na sljedećim katovima: 8, 9, 28, 29, 48, 49, 88 i 89. Proučavajući još par slučajeva možemo se uvjeriti da će biti uvijek broj mogućnosti za prvi ekran pomnožen s brojem mogućnosti za drugi ekran. Potrebno je bilo paziti još na slučaj kada je s ekranom moguće biti i na nultom katu. Budući da nulti kat ne postoji, ne smijemo taj slučaj brojiti.

Programski kod (pisan u Python 3)

```
nula=1
def znamenke(x):
    global nula
    br=1
    if 'G' not in x:
        br+=1
    else:
        nula=0
```

```
    if 'A' not in x and 'F' not in x and 'G' not in x and 'E' not in x and 'D'
not in x:
        br+=1
    if 'F' not in x and 'C' not in x:
        br+=1
    if 'F' not in x and 'E' not in x:
        br+=1
    if 'A' not in x and 'E' not in x and 'D' not in x:
        br+=1
    if 'B' not in x and 'E' not in x:
        br+=1
    if 'B' not in x:
        br+=1
    if 'F' not in x and 'G' not in x and 'E' not in x and 'D' not in x:
        br+=1
    if 'E' not in x:
        br+=1
    return br
n = int(input())
sol = 1
for i in range(n):
    x = input()
    sol *= znamenke(x)
print(sol - nula)
```

Potrebno znanje: naredba odlučivanja

Kategorija: ad hoc

7.1. Zadatak: Podjela Autor:

Vidi 5.3

7.2. Zadatak: Lift Autor:

Vidi 6.3

7.3. Zadatak: Zmija Autor: Stjepan Požgaj

Implementacijom koraka iz teksta zadatka dolazimo do rješenja.

Programski kod (pisan u C++)

```
#include <iostream>
```

```
#include <cstdio>
#include <deque>
#include <string>
using namespace std;
#define TRACE(x) cerr << #x << " = " << x << endl
#define FOR(i, a, b) for(int i = (a); i < (b); i++)
#define REP(i, n) FOR(i, 0, n)
typedef long long int llint;
typedef pair<int, int> par;
#define X first
#define Y second
const int MAXN = 110;
deque<par> zmija;

int n, m, a, b, k, l;
int mat[MAXN][MAXN];
string s;
bool zauzeto[MAXN][MAXN];
void unos() {
  cin >> n >> m;
  REP(i, n)
    REP(j, m)
      cin >> mat[i][j];
  cin >> a >> b;
  a--, b--;
  cin >> l;
  cin >> s;
  cin >> k;
}
par pomak(par pos, char c) {
  if(c == 'U')
    return par((pos.X - 1 + n) % n, pos.Y);
  else if(c == 'D')
    return par((pos.X + 1) % n, pos.Y);
  else if(c == 'L')
    return par(pos.X, (pos.Y - 1 + m) % m);
```

```
  else
    return par(pos.X, (pos.Y + 1) % m);
}
int main() {
  unos();
  int it = 0;
  bool ugrizao = false;
  zmija.push_front(par(a, b));
  zauzeto[a][b] = true;
  FOR(i, 1, k) {
    par glava = zmija.front();
    par novi = pomak(glava, s[it]);
    if(!mat[glava.X][glava.Y]) {
      par rep = zmija.back();
      zauzeto[rep.X][rep.Y] = false;
      zmija.pop_back();
    }
    if(zauzeto[novi.X][novi.Y])
      ugrizao = true;
    zmija.push_front(novi);
    zauzeto[novi.X][novi.Y] = true;
    if(ugrizao) break;
    it = (it + 1) % l;
  }
  if(ugrizao)
    cout << "DA" << endl;
  else
    cout << "NE" << endl;
  par glava = zmija.front();
  cout << glava.X + 1 << " " << glava.Y + 1 << endl;
  par rep = zmija.back();
  cout << rep.X + 1 << " " << rep.Y + 1 << endl;
  return 0;
}
```

Potrebno znanje: dvodimenzionalna polja, simulacija

Kategorija: ad hoc

8.1. Zadatak: Prekid Autor:

Vidi 6.2

8.2. Zadatak: Zmija Autor:

Vidi 7.3

8.3. Zadatak: Brodovi Autor: Patrick Pavić

U zadatku zapravo želimo popločati 8 polja s dvije 1x4 domine tako da minimalna vrijednost koju smo popločali je maksimalna moguća. Za svako polje izračunamo najmanju vrijednost koju smo popločili ako lijevi kraj domine postavimo na to polje. Nakon toga sortiramo polja prema toj vrijednosti. Sada možemo primijetiti da ćemo za postavljanje dvije domine izabrati neka dva od prvih par polja, npr. 20, (koliko točno treba uzeti ostavljamo čitatelju za vježbu) u tom sortiranom nizu, trebamo samo pripaziti da se polja ne sijeku.

Programski kod (pisan u C++)

```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef double lf;
typedef long double Lf;
typedef pair <int,int> pii;
typedef pair <ll, ll> pll;

#define TRACE(x) cerr << #x << "  " << x << endl
#define FOR(i, a, b) for (int i = (a); i < int(b); i++)
#define REP(i, n) FOR(i, 0, n)
#define all(x) (x).begin(), (x).end()
#define _ << " " <<

#define fi first
#define sec second
#define mp make_pair
#define pb push_back

int n, m;

vector <pair <int, pii> > v;
```

```
int main() {
  cin >> n >> m;
  vector <vector <int> > p(n);
  for (auto &x : p) {
    x.resize(m);
  }
  REP(i, n) {
    REP(j, m) {
      cin >> p[i][j];
    }
  }

  REP(i, n) {
    REP(j, m - 3) {
      int t = p[i][j];
      REP(k, 4) {
        t = min(t, p[i][j + k]);
      }
      v.pb({t, {i, j}});
    }
  }
  sort(all(v));
  reverse(all(v));

  int ans = 0;
  REP(i, min(20, (int)v.size())) {
    FOR(j, i + 1, min(20, (int)v.size())) {
      int a = v[i].sec.fi;
      int b = v[i].sec.sec;
      int c = v[j].sec.fi;
      int d = v[j].sec.sec;
      if (a != c || abs(b - d) >= 4) {
        ans = max(ans, min(v[i].fi, v[j].fi));
      }
    }
  }
```

```
  cout << ans << endl;
  return 0;
}
```

Potrebno znanje:

Kategorija: ad hoc