ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΑΘΛΗΤΙΣΜΟΥ ΚΑΙ ΝΕΟΛΑΙΑΣ

ΔΙΕΥΘΥΝΣΗ ΑΝΩΤΕΡΗΣ ΕΚΠΑΙΔΕΥΣΗΣ

ΥΠΗΡΕΣΙΑ ΕΞΕΤΑΣΕΩΝ

ΠΑΓΚΥΠΡΙΕΣ ΕΞΕΤΑΣΕΙΣ ΠΡΟΣΒΑΣΗΣ 2026

ΠΡΟΤΕΙΝΟΜΕΝΕΣ ΛΥΣΕΙΣ

Μάθημα: ΠΛΗΡΟΦΟΡΙΚΗ (15)

Ημερομηνία και ώρα εξέτασης: Τετάρτη, 17 Ιουνίου 2026

08:00 – 11:00

ΟΙ ΠΡΟΤΕΙΝΟΜΕΝΕΣ ΛΥΣΕΙΣ ΑΠΟΤΕΛΟΥΝΤΑΙ ΑΠΟ ΕΙΚΟΣΙΟΚΤΩ (28) ΣΕΛΙΔΕΣ

ΜΕΡΟΣ Α΄

ΑΣΚΗΣΗ 1:

Δίνεται το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++:

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;
int main(){
int n1,n2,n3,n4;
float r1,r2;
cin >> n1 >> n2 >> n3;
```

Α

```
if(
```

Β

```
){
```

Γ

Δ

```
}
```

Ε

```
return 0;
}
```

Να γράψετε στο τετράδιο απαντήσεών σας:

(α) Την εντολή που πρέπει να τοποθετηθεί στη **θέση Α**, έτσι ώστε να εκχωρείται στην μεταβλητή **r1** ο μέσος όρος των τιμών των μεταβλητών n1, n2 και n3.

(Μονάδα 1)

(β) Τη συνθήκη που πρέπει να τοποθετηθεί στη **θέση Β**, η οποία ελέγχει αν η μεταβλητή **n1** είναι άνιση με την μεταβλητή **n2** ή αν η μεταβλητή **n1** είναι ίση με την μεταβλητή **n3**.

(Μονάδα 1)

(γ) Την εντολή που πρέπει να τοποθετηθεί στη **θέση Γ**, έτσι ώστε να εκχωρείται στην μεταβλητή **n4** το άθροισμα των ψηφίων των μονάδων των μεταβλητών n1 και n2.

(Μονάδα 1)

(δ) Την εντολή που πρέπει να τοποθετηθεί στη **θέση Δ**, έτσι ώστε να εκχωρείται στην μεταβλητή **r2** το ακέραιο μέρος της μεταβλητής **r1**.

(Μονάδα 1)

(ε) Την εντολή που πρέπει να τοποθετηθεί στη **θέση Ε**, έτσι ώστε να τυπώνεται η τιμή της μεταβλητής **r1** με ακρίβεια δύο (2) δεκαδικών ψηφίων.

(Μονάδα 1)

Λύσεις:

(α) `r1=(n1+n2+n3)/3.0;` ή `r1=(float)((n1+n2+n3)/3);`

(β) `n1!=n2 || n1==n3`

(γ) `n4=(n1+n2)%10;`

(δ) `r2=trunc(r1);` ή `r2=int(r1);`

(ε) `cout<<fixed<<setprecision(2)<<r1;`

ΑΣΚΗΣΗ 2:

Δίνονται στο **δεκαδικό** σύστημα αρίθμησης ο πραγματικός αριθμός **Α=38.25** και στο **δυαδικό** σύστημα αρίθμησης οι αριθμοί **Β=10001010** και **Γ=00101000**.

(α) Να δείξετε ότι η αντίστοιχη τιμή του **πραγματικού αριθμού Α** στο δυαδικό σύστημα αρίθμησης είναι **$(100110.01)_2$,** σημειώνοντας τα βήματα που ακολουθήσατε για να φτάσετε στο συγκεκριμένο αποτέλεσμα.

(Μονάδες 2)

(β) Να δείξετε ότι η αντίστοιχη τιμή του **δυαδικού αριθμού Β** στο δεκαδικό σύστημα αρίθμησης είναι **$(138)_{10}$**, σημειώνοντας τα βήματα που ακολουθήσατε για να φτάσετε στο συγκεκριμένο αποτέλεσμα.

(Μονάδα 1)

(γ) Χρησιμοποιώντας το **συμπλήρωμα ως προς 2** του δυαδικού αριθμού **Γ**, να υπολογίσετε στο δυαδικό σύστημα αρίθμησης, το αποτέλεσμα της **αφαίρεσης Β-Γ**.

(Μονάδες 2)

Λύσεις:

(α) Α=38.25

Ακέραιο μέρος			**Δεκαδικό μέρος**		
Αριθμός	Πηλίκο	Υπόλοιπο	Αριθμός	Γινόμενο	Bit
38/2=	19	0 ↑	0.25*2	0.5	0
19/2=	9	1	0.5*2	1	1 ↓
9/2=	4	1			
4/2=	2	0			
2/2=	1	0			
1/2=	0	1			

(β) Β=10001010 Β=128+8+2=$(138)_{10}$

(γ) Β=10001010 Γ=00101000 Συμπλήρωμα Γ ως προς 2: 11011000

10001010
11011000+
101100010

ΑΣΚΗΣΗ 3

(α) Δίνεται το πιο κάτω πρόγραμμα στη γλώσσα προγραμματισμού C++. Χρησιμοποιώντας οποιονδήποτε τρόπο (π.χ. μέθοδο της προκαταρκτικής εκτέλεσης), να γράψετε τα αποτελέσματα του προγράμματος, όπως θα τυπωθούν στην οθόνη.

```
#include<iostream>
using namespace std;

int syn3a(int a, int &b){
    int c = 0;
    for(int i=a; i<=b; i+=2)
        c += i;
    a++;
    b++;
return c;
}
int main(){
    int x = 2, y = 7, z;
    z = syn3a(x,y);
    cout << x << endl;
    cout << y << endl;
    cout << z;
return 0;
}
```

(Μονάδες 3)

(β) Το πιο κάτω πρόγραμμα, στη γλώσσα προγραμματισμού C++, διαβάζει από το πληκτρολόγιο τη συμβολοσειρά **stA**. Στη συνέχεια καλεί τη συνάρτηση **syn3b** η οποία δέχεται ως παράμετρο τη συμβολοσειρά stA. Η συνάρτηση syn3b υπολογίζει και επιστρέφει στην κύρια συνάρτηση (**main**) το πλήθος των κεφαλαίων χαρακτήρων του λατινικού αλφαβήτου που υπάρχουν μέσα στη συμβολοσειρά stA. Η επιστρεφόμενη τιμή τυπώνεται στην κύρια συνάρτηση.

Στο πρόγραμμα υπάρχουν λογικά ή/και συντακτικά λάθη. Να γράψετε στο τετράδιο απαντήσεών σας **τέσσερα (4)** από αυτά, αναφέροντας τον αριθμό της γραμμής στην οποία εμφανίζεται το κάθε λάθος μαζί με τη διορθωμένη εντολή. Στο πρόγραμμα **να μην γίνει κάποια προσθήκη ή αφαίρεση εντολής**.

```
/*1*/  #include <iostream>
/*2*/  #include <string>
/*3*/  using namespace std;
/*4*/  int syn3b(string a){
/*5*/    int p;
/*6*/    for(int i=0; i<=a.size(); i++)
/*7*/         if(a[i]>='A' || a[i]<='Z')
/*8*/             p++;
/*9*/    return p;
/*10*/  }
/*11*/  int main(){
/*12*/    string stA;
/*13*/    cin >> stA;
/*14*/    cout << syn3b(string stA);
/*15*/  return 0;
/*16*/  }
```

(Μονάδες 2)

Λύσεις:

(α)

2

8

12

(β)

```
/*5*/    int p=0;
/*6*/    for (int i=0; i<a.size(); i++)
/*7*/    if (a[i]>= 'A ' && a[i]<= 'Z')
/*14*/   cout <<syn3b(stA);
```

ΑΣΚΗΣΗ 4:

Μία έγκυρη διεύθυνση ηλεκτρονικού ταχυδρομείου (email) έχει τη μορφή όνομα_χρήστη@τομέας. Για παράδειγμα, το όνομα χρήστη στο email student@schools.ac.cy είναι η συμβολοσειρά student.

Να γράψετε πρόγραμμα, στη γλώσσα προγραμματισμού C++, το οποίο να διαβάζει από το αρχείο **accounts.txt** τα emails άγνωστου πλήθους φοιτητών και στη συνέχεια να τυπώνει, σε διαφορετικές γραμμές, στο αρχείο **userscodes.txt** τα ονόματα των χρηστών (usernames).

(Μονάδες 5)

Παράδειγμα Εισόδου (accounts.txt)	**Παράδειγμα Εξόδου (userscodes.txt)**
john@st.schools.ac.cy mary@cs.ouc.cy nick@cs.ucy.cy	john mary nick

Λύση:

```
#include<fstream>
using namespace std;

int main(){
    ifstream fin("accounts.txt");
    ofstream fout("userscodes.txt");
    string str, user;
    int i;
    while(!fin.eof()){
        fin >> str;
        i=0;
        user.clear();
        while(str[i]!='@'){
            user += str[i];
            i++;
        }
        fout << user << endl;
    }
    fin.close();
    fout.close();
return 0;
}
```

ΑΣΚΗΣΗ 5:

Το παραγοντικό ενός θετικού ακέραιου αριθμού Ν, συμβολίζεται με Ν! και είναι το γινόμενο όλων των ακεραίων από το 1 μέχρι και το Ν. Για παράδειγμα: 4!=1*2*3*4=24.

Να γράψετε πρόγραμμα, στη γλώσσα προγραμματισμού C++, το οποίο να:

(α) **υπολογίζει** το παραγοντικό του κάθε αριθμού από το **1** μέχρι και το **6** και να το καταχωρίζει σε έναν μονοδιάστατο πίνακα με όνομα **F**.

(Μονάδες 2)

(β) διαβάζει έναν θετικό ακέραιο αριθμό **N (1≤N<1000)** και χρησιμοποιώντας τις τιμές του πίνακα F να εκφράζει τον αριθμό Ν ως άθροισμα παραγοντικών, επιλέγοντας σε κάθε βήμα **το μεγαλύτερο δυνατό** παραγοντικό από τον πίνακα. Για κάθε παραγοντικό που συμμετέχει στο άθροισμα, να εμφανίζει τον συντελεστή (δηλαδή πόσες φορές χρησιμοποιείται το κάθε παραγοντικό) και το αντίστοιχο παραγοντικό του.

Σημείωση: Να τυπώνονται μόνο τα παραγοντικά που έχουν μη μηδενικούς συντελεστές.

(Μονάδες 3)

Παράδειγμα Εισόδου	**Παράδειγμα Εξόδου**	**Επεξήγηση**
13	2*3! 1*1!	2*3!+1*1! = 12 + 1 = 13
181	1*5! 2*4! 2*3! 1*1!	1*5!+2*4!+2*3!+1*1!= 120 + 48 + 12 + 1 = 181

Λύση:

```
#include<iostream>
using namespace std;
int main(){
    int N,p,F[6],d;
    cin>>N;
    for(int i=1;i<=6;i++){
        p=1;
        for(int j=1;j<=i;j++)
            p*=j;
        F[i-1]=p;
    }
```

```
    for(int i=5;i>=0;i--){
        d=N/F[i];
        if(d!=0){
            N=N%F[i];
            cout<<d<<'*'<<i+1<<'!'<<endl;
        }
    }
return 0;
}
```

ΑΣΚΗΣΗ 6:

Να γράψετε συνάρτηση, στη γλώσσα προγραμματισμού C++, με πρότυπο:

```
int bsearch(int N, string W[], string target);
```

Η συνάρτηση δέχεται ως παραμέτρους:

- έναν ακέραιο αριθμό **N**, ο οποίος αντιστοιχεί στο πλήθος των συμβολοσειρών του πίνακα
- έναν πίνακα **W**, ο οποίος περιέχει **N** διαφορετικές συμβολοσειρές, σε αλφαβητική σειρά
- μια συμβολοσειρά με όνομα **target** η οποία δεν υπάρχει στον πίνακα W

Η συνάρτηση να χρησιμοποιεί τον αλγόριθμο **Δυαδικής Αναζήτησης (Binary Search)** και να επιστρέφει τη θέση του πίνακα **μετά την οποία** πρέπει να εισαχθεί η συμβολοσειρά **target**, ώστε ο πίνακας να παραμείνει ταξινομημένος.

Αν η συμβολοσειρά **target** πρέπει να τοποθετηθεί πριν από το πρώτο στοιχείο του πίνακα, η συνάρτηση πρέπει να επιστρέφει την τιμή -1.

Παράδειγμα:

Παράμετροι Συνάρτησης	**Επιστρέφει**	**Επεξήγηση**
N = 3 Πίνακας W: "aa","abc","dog" target: "add"	1	Η συμβολοσειρά "add" πρέπει να τοποθετηθεί μετά τη λέξη "abc" που βρίσκεται στη θέση 1.
N = 3 Πίνακας W: "cat","dog","fox" target: "bat"	-1	Η συμβολοσειρά "bat" πρέπει να τοποθετηθεί πριν από το πρώτο στοιχείο του πίνακα.

(Μονάδες 5)

Λύση:

```
int bsearch(int N, string W[], string target){
    int mid,first=0,last=N-1;
    while(first<=last){
        mid=(first+last)/2;
        if(target>W[mid])
            first=mid+1;
        else
            last=mid-1;
    }
    return last;
}
```

- ΤΕΛΟΣ Α΄ ΜΕΡΟΥΣ -
- ΑΚΟΛΟΥΘΕΙ ΤΟ ΜΕΡΟΣ Β΄-

ΜΕΡΟΣ Β΄

ΑΣΚΗΣΗ 7:

(α) Το **Σύστημα Ανίχνευσης Κόπωσης Οδηγού** ενεργοποιεί σταδιακή επιβράδυνση του αυτοκινήτου μέχρι την πλήρη ακινητοποίησή του. Το σύστημα χρησιμοποιεί τους πιο κάτω αισθητήρες:

- **Α - Χρόνος Οδήγησης:** βρίσκεται στην κατάσταση 1 όταν ο οδηγός οδηγεί πάνω από τέσσερις συνεχόμενες ώρες, διαφορετικά βρίσκεται στην κατάσταση 0.
- **Β - Κάμερα βλεφάρων:** βρίσκεται στην κατάσταση 1 όταν τα βλέφαρα του οδηγού είναι κλειστά για πάνω από τρία δευτερόλεπτα, διαφορετικά βρίσκεται στην κατάσταση 0.
- **C - Αισθητήρας τιμονιού:** βρίσκεται στην κατάσταση 1 όταν ο οδηγός δεν κρατάει το τιμόνι με τα δύο χέρια, διαφορετικά βρίσκεται στην κατάσταση 0.

Το σύστημα ενεργοποιείται (**F=1)** όταν ισχύει **τουλάχιστον μια** από τις ακόλουθες συνθήκες:

- Ο οδηγός οδηγεί πάνω από τέσσερις συνεχόμενες ώρες και τα βλέφαρα του είναι κλειστά για πάνω από τρία δευτερόλεπτα.
- Ο οδηγός **ΔΕΝ** κρατάει το τιμόνι με τα δύο χέρια.

Να δημιουργήσετε **τον πίνακα αληθείας** για το πιο πάνω αυτοματοποιημένο σύστημα και να γράψετε την αντίστοιχη **λογική συνάρτηση F** του συστήματος **σε μορφή αθροίσματος ελαχιστόρων.**

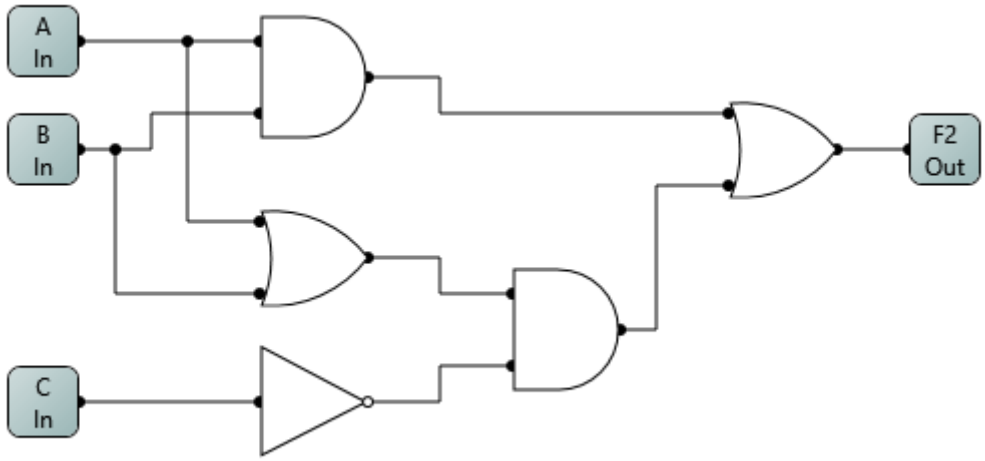
(Μονάδες 5)

(β) Δίνεται ο πιο κάτω χάρτης **Karnaugh** τεσσάρων μεταβλητών. Αφού τον αντιγράψετε στο τετράδιο απαντήσεών σας, να ομαδοποιήσετε τους γειτονικούς του όρους και να γράψετε τη λογική συνάρτηση **F1(A, B, C, D)** που προκύπτει στην πιο απλή της μορφή.

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	1	1	0	0

(Μονάδες 3)

(γ) Να γράψετε τη λογική συνάρτηση **F2** που αντιστοιχεί στο πιο κάτω λογικό κύκλωμα και να υπολογίσετε την τιμή της **F2** αν **A=1, B=0** και **C=0.**



(Μονάδες 2)

Λύσεις:

(α)

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

F(A,B,C)=A'B'C +A'BC +AB'C +ABC'+ABC

Ή

F(A,B,C)=Σ(1, 3, 5, 6, 7)

(β)

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	1	1	0	0
10	1	1	0	0

F(A,B,C,D)=A'D+AC'

(γ) F2(A,B,C)=AB+(A+B)C'

F2=1*0+(1+0)*1=0+1=1

ΑΣΚΗΣΗ 8:

Δίνεται μια συμβολοσειρά **st** που αποτελείται το πολύ από **1000** λέξεις. Οι λέξεις χωρίζονται με ένα κενό διάστημα. Μετά την τελευταία λέξη δεν υπάρχει κενό διάστημα.

Να γράψετε στη γλώσσα προγραμματισμού C++, τρεις (3) συναρτήσεις όπως αυτές περιγράφονται πιο κάτω, θεωρώντας ότι υπάρχουν δηλωμένες οι απαραίτητες βιβλιοθήκες:

(α) Τη συνάρτηση **count**, η οποία δέχεται ως παράμετρο τη συμβολοσειρά st και επιστρέφει το πλήθος των λέξεών της.

Παράδειγμα (συμβολοσειρά st)	**Επιστρέφει**
`modify this text for the students`	`6`

(Μονάδες 2)

(β) Τη συνάρτηση **lexeis,** η οποία δέχεται ως παράμετρο τη συμβολοσειρά st και έναν πίνακα συμβολοσειρών με όνομα **words**. Η συνάρτηση χωρίζει τη συμβολοσειρά σε λέξεις και τις αποθηκεύει στον πίνακα words.

Παράδειγμα (συμβολοσειρά st)	**Περιεχόμενα πίνακα words**

`modify this text for the students`	`modify`
	`this`
	`text`
	`for`
	`the`
	`students`

(Μονάδες 4)

(γ) Τη συνάρτηση **align,** η οποία δέχεται τις ακόλουθες παραμέτρους:

- τον πίνακα συμβολοσειρών **words** που περιέχει τις λέξεις της συμβολοσειράς
- έναν ακέραιο αριθμό **N** που αντιστοιχεί στο πλήθος των λέξεων που υπάρχουν μέσα στον πίνακα words
- έναν ακέραιο αριθμό **X** που αντιστοιχεί στο μέγιστο πλήθος χαρακτήρων που μπορούν να τυπωθούν σε μια γραμμή

Η συνάρτηση **τυπώνει στην οθόνη** σε γραμμές μεγέθους μέχρι X χαρακτήρες τις λέξεις που υπάρχουν μέσα στον πίνακα. Να θεωρήσετε ότι η κάθε λέξη αποτελείται το πολύ από X χαρακτήρες. Οι λέξεις τυπώνονται με αριστερή στοίχιση και χωρίζονται μεταξύ τους από έναν κενό χαρακτήρα.

Παράδειγμα			
Πίνακας words	**Ακέραιος N**	**Ακέραιος X**	**Οθόνη**

`modify`
`this`
`text`
`for`
`the`
`students`
	`6`	`11`	`modify this`
			`text for`
			`the`
			`students`

(Μονάδες 4)

Λύση:

```
int count(string st){
  int ans=1;
  for(int i=0; i<st.size(); i++){
      if(st[i]==' ')
        ans++;
  }
return ans;
}

void lexeis(string st, string words[]){
   string word;
   int cnt=0;
   for(int i=0; i<st.size(); i++){
       if(st[i]==' '){
           words[cnt]=word;
           cnt++;
           word.clear();
       }
        else
            word+=st[i];
       }
    words[cnt]=word;
}

void align(string words[], int N, int X) {
    string line=words[0];
    for(int i=1; i<N; i++)
          if(line.size() + words[i].size() + 1 <= X){
            line+=' ';
            line+=words[i];
          }
          else{
            cout<<line<<endl;
            line=words[i];
          }

    if(!line.empty())
        cout<<line<<endl;
}
```

ΑΣΚΗΣΗ 9:

Η διεύθυνση ενός ταξιδιωτικού γραφείου, επιθυμεί τη δημιουργία προγράμματος, στη γλώσσα προγραμματισμού C++, το οποίο να:

(α) ορίζει μια **εγγραφή** με το όνομα **sights**, η οποία να έχει τα παρακάτω μέλη:

- **name**: το όνομα του αξιοθέατου (συμβολοσειρά χωρίς κενά)
- **town**: την πόλη που βρίσκεται το αξιοθέατο (συμβολοσειρά χωρίς κενά)
- **V**: αριθμός επισκεπτών τα τελευταία επτά (7) χρόνια (πίνακας ακεραίων)

Στη συνέχεια να **ορίζει** έναν **πίνακα εγγραφών** τύπου **sights**, με όνομα **S**, πενήντα (50) θέσεων.

(Μονάδες 2)

(β) διαβάζει τα στοιχεία: όνομα αξιοθέατου, πόλη και αριθμός επισκεπτών τα τελευταία 7 χρόνια για κάθε αξιοθέατο και καταχωρίζει τα στοιχεία αυτά στον πίνακα εγγραφών **S**. Η είσοδος δεδομένων τερματίζεται όταν ο χρήστης καταχωρίσει στοιχεία για 50 αξιοθέατα ή όταν δώσει ως όνομα αξιοθέατου τη λέξη «**EXIT**».

(Μονάδες 4)

(γ) εμφανίζει στην οθόνη το **όνομα** και την **πόλη** των αξιοθέατων για τα οποία ο αριθμός των επισκεπτών παρουσιάζει συνεχή ανοδική πορεία τα τελευταία 7 χρόνια. Δηλαδή, ο αριθμός των επισκεπτών κάθε χρονιάς είναι μεγαλύτερος από τον αντίστοιχο της αμέσως προηγούμενης χρονιάς.

Παράδειγμα Εισόδου	**Παράδειγμα Εξόδου**
Κάστρο_Λεμεσού Λεμεσός 1200 1350 1410 1500 1620 1750 1900 Τάφοι_των_Βασιλέων Πάφος 2500 2600 2550 2700 2850 3000 3150 Μουσείο_Κύπρου Λευκωσία 1800 1950 2100 2250 2400 2600 2800 EXIT	Κάστρο_Λεμεσού - Λεμεσός Μουσείο_Κύπρου - Λευκωσία

Λύση:

```
#include <iostream>
#include <string>
using namespace std;
struct sights{
     string name, town;
     int V[7];
}S[50];
```

```
int main(){

int i=0,j, N;
bool f;
cin>>S[i].name;
while(S[i].name!="EXIT" && i<50){

    cin>>S[i].town;
    for(j=0;j<7;j++)
        cin>>S[i].V[j];
    i++;
    cin>>S[i].name;
}



N=i;
for(i=0;i<N;i++){
    f=true;
    for(j=0;j<6;j++)
        if (S[i].V[j]>=S[i].V[j+1])
            f=false;
    if(f)
        cout<<S[i].name<<"-"<<S[i].town<<endl;

}
return 0;
}
```

ΑΣΚΗΣΗ 10:

Οι γονείς μιας μαθήτριας της χάρισαν **Κ** κουτιά με σοκολάτες. Κάθε κουτί περιέχει διαφορετικό είδος σοκολατών. Δύο ή περισσότερα κουτιά μπορεί να περιέχουν την ίδια ποσότητα. Κάθε κουτί περιέχει από 1 μέχρι και 50 σοκολάτες. Η μαθήτρια τρώει μία σοκολάτα την ημέρα για τις πέντε (5) μέρες που πηγαίνει στο σχολείο (Δευτέρα έως Παρασκευή).

Να γράψετε πρόγραμμα στη γλώσσα προγραμματισμού C++, το οποίο να:

(α) διαβάζει έναν ακέραιο αριθμό **Κ (1≤Κ≤500)** ο οποίος αντιπροσωπεύει το πλήθος των κουτιών σοκολάτας. Ακολούθως, να διαβάζει το πλήθος σοκολατών για κάθε ένα από τα **Κ** κουτιά και να τα καταχωρίζει στον πίνακα **chocos**.

(Μονάδες 2)

(β) εντοπίζει και τυπώνει την ποσότητα σοκολατών (**F**) που εμφανίζεται τις περισσότερες φορές αρχικά στα κουτιά. Να θεωρήσετε ότι υπάρχει μόνο μία τέτοια ποσότητα.

(Μονάδες 3)

(γ) υπολογίζει και τυπώνει τον συνολικό αριθμό ημερών (**D**) κατά τις οποίες η μαθήτρια τρώει σοκολάτες, δεδομένου ότι ακολουθεί τους εξής κανόνες:

- Κάθε μέρα τρώει ακριβώς μία σοκολάτα.
- Κατά τη διάρκεια μιας εβδομάδας (**πέντε μέρες**) δεν μπορεί να φάει σοκολάτα από το ίδιο κουτί. Σε διαφορετικές εβδομάδες μπορεί να φάει ξανά σοκολάτα από το ίδιο κουτί.
- Η μαθήτρια επιλέγει τις σοκολάτες που τρώει με τον καλύτερο δυνατό τρόπο, ώστε το πλήθος των ημερών να είναι το μέγιστο δυνατό, δηλαδή να φάει όσον το δυνατό περισσότερες σοκολάτες.
- Σταματά να τρώει σοκολάτες την ημέρα κατά την οποία δεν υπάρχει διαθέσιμο διαφορετικό είδος σοκολάτας για να φάει, εντός της ίδιας εβδομάδας.

(Μονάδες 5)

Παράδειγμα Εισόδου	**Παράδειγμα Εξόδου**	**Επεξήγηση**
5 1 1 1 1 2	F: 1 D: 6	Η ποσότητα που εμφανίζεται τις περισσότερες φορές είναι η **1** (4 φορές). Την 1η εβδομάδα τρώει από τα 5 διαφορετικά κουτιά σοκολατών. Τη 2η εβδομάδα μόνο το 5ο κουτί έχει ακόμη διαθέσιμη ποσότητα, άρα τρώει ακόμη μια σοκολάτα.
6 3 5 3 2 4 1	F: 3 D: 17	Η ποσότητα που εμφανίζεται τις περισσότερες φορές είναι η **3** (2 φορές). Με βέλτιστη επιλογή σοκολατών: Τρεις εβδομάδες από 5 διαφορετικά κουτιά σοκολατών και την 4η εβδομάδα άλλες 2 σοκολάτες.
4 30 30 30 50	F: 30 D: 4	Η ποσότητα που εμφανίζεται τις περισσότερες φορές είναι η **30** (3 φορές). Υπάρχουν μόνο 4 διαφορετικά κουτιά σοκολατών, οπότε την 5η ημέρα της 1ης εβδομάδας σταματά γιατί δεν υπάρχει διαθέσιμο άλλο είδος σοκολάτας.

Ανάλυση λύσης

Στο **ερώτημα (α)** διαβάζεται αρχικά το πλήθος των κουτιών K και στη συνέχεια οι ποσότητες σοκολάτας κάθε κουτιού αποθηκεύονται στον πίνακα chocos. Κάθε θέση του πίνακα αντιστοιχεί σε διαφορετικό κουτί, άρα ακόμη και αν δύο κουτιά έχουν την ίδια ποσότητα, θεωρούνται διαφορετικά είδη.

Στο **ερώτημα (β)** χρησιμοποιείται πίνακας συχνοτήτων, αφού κάθε ποσότητα είναι από 1 μέχρι 50. Για κάθε τιμή chocos[i] αυξάνεται η αντίστοιχη θέση του πίνακα συχνοτήτων. Στη συνέχεια εντοπίζεται η ποσότητα με τη μεγαλύτερη συχνότητα και εμφανίζεται ως F.

Στο **ερώτημα (γ)** χρειάζεται να μεγιστοποιηθεί ο αριθμός ημερών D. Σε κάθε εργάσιμη εβδομάδα η μαθήτρια μπορεί να φάει μέχρι 5 σοκολάτες, αλλά κάθε μία πρέπει να προέρχεται από διαφορετικό κουτί.

Για να πετύχουμε το μέγιστο δυνατό πλήθος ημερών, σε κάθε εβδομάδα επιλέγουμε τα κουτιά με τις μεγαλύτερες διαθέσιμες ποσότητες. Έτσι αποφεύγουμε να εξαντληθούν γρήγορα πολλά κουτιά και κρατούμε όσο γίνεται περισσότερες επιλογές για τις επόμενες εβδομάδες.

Στη Λύση 1 αυτό γίνεται ταξινομώντας κάθε φορά τον πίνακα chocos σε φθίνουσα σειρά. Αν υπάρχουν τουλάχιστον 5 κουτιά με διαθέσιμες σοκολάτες, αφαιρείται μία σοκολάτα από τα 5 πρώτα κουτιά και προστίθενται 5 ημέρες. Αν υπάρχουν λιγότερα από 5 κουτιά με διαθέσιμες σοκολάτες, προστίθενται μόνο αυτές οι ημέρες και η διαδικασία σταματά.

Στη Λύση 2 η ίδια λογική υλοποιείται χωρίς ταξινόμηση. Για κάθε εβδομάδα χρησιμοποιείται βοηθητικός πίνακας B, ώστε να σημειώνεται ποια κουτιά έχουν ήδη χρησιμοποιηθεί μέσα στην ίδια εβδομάδα. Κάθε μέρα επιλέγεται το διαθέσιμο μη χρησιμοποιημένο κουτί με τη μεγαλύτερη ποσότητα. Αν δεν μπορεί να επιλεγεί νέο διαφορετικό κουτί, η εβδομάδα σταματά και μαζί σταματά η συνολική διαδικασία.

Και οι δύο λύσεις εφαρμόζουν την ίδια greedy ιδέα: σε κάθε βήμα επιλέγεται κουτί με όσο το δυνατό μεγαλύτερη διαθέσιμη ποσότητα, χωρίς να επαναλαμβάνεται το ίδιο κουτί στην ίδια εβδομάδα.

Λύση 1:

```
#include <iostream>
using namespace std;

 void isort(int arr[], int N){
   int temp,j;
   for(int i=1; i<N; i++){
       temp=arr[i];
       j=i-1;
       while(j>=0 && arr[j]<temp){
           arr[j+1] = arr[j];
           j--;
```

```
        }
    arr[j+1]=temp;
   }
}
int main() {
    int chocos[500], occ[51]={0}, K, days=0, cnt, maxf=0, pos;
    bool check = true;
    cin >> K;
    for (int i = 0; i < K; i++) {
        cin >> chocos[i];
        occ[chocos[i]]++;
    }
    for(int i=0; i<51; i++)
        if(occ[i]>maxf){
            maxf=occ[i];
            pos=i;
        }
    while(check) {
        isort(chocos, K);
        cnt = 0;
        while (cnt < K && chocos[cnt]>0)
            cnt++;
        if (cnt<5){
            days+=cnt;
            check=false;
        }
        else{
            for(int i=0; i<5; i++) {
                chocos[i]--;
            }
            days += 5;
        }
    }
    cout << "F: " << pos << endl;
    cout << "D: " << days;
return 0;
}
```

Λύση 2:

```
#include<iostream>
using namespace std;
int main(){
    int chocos[500],K,F[51]={},max=0,pos;
    //erotima A
    cin>>K;
    for(int i=0;i<K;i++)
        cin>>chocos[i];
```

```
    //erotima B
    for(int i=0;i<K;i++)
        F[chocos[i]]++;
    for(int i=1;i<=50;i++)
        if(F[i]>max){
            max=F[i];
            pos=i;
        }
    cout<<"F: "<<pos<<endl;

    //erotima C
    bool check=true,B[500];
    int counter,total=0,j;
    while(check){
        counter=0;
        j=0;
        for(int q=0;q<K;q++)
            B[q]=false;
        while(counter<5 && j<K){
            max=0;
            for(int q=0;q<K;q++)
            if(max<chocos[q] && !B[q]){
                max=chocos[q];
                pos=q;
            }
            if(max!=0){
                chocos[pos]--;
                counter++;
                B[pos]=true;
            }
            j++;
        }
        if(counter!=5)
            check=false;
        total+=counter;
    }
    cout<<"D: "<<total;
return 0;
}
```

- ΤΕΛΟΣ Β΄ ΜΕΡΟΥΣ -
- ΑΚΟΛΟΥΘΕΙ ΤΟ ΜΕΡΟΣ Γ΄-

ΜΕΡΟΣ Γ΄

Άσκηση 11:

Μια ομάδα χωραφιών είναι διατεταγμένη σε ένα δισδιάστατο πλέγμα με **N** γραμμές και **M** στήλες (**5≤N<1000, 5≤M<1000**). Μετά από έντονη βροχόπτωση, ορισμένα χωράφια έχουν πλημμυρίσει. Σε κάθε γραμμή αρχικά μπορεί να υπάρχει το πολύ ένα πλημμυρισμένο χωράφι.

Δίνεται το πιο κάτω τμήμα προγράμματος στη γλώσσα προγραμματισμού C++:

```
int main(){
    char A[1000][1000];
    int B[1000],F[1000],N,M;
    cin >> N >> M;
    fill(A,N,M);
    calc(A,B,F,N,M);
    cout << "D: "<< dry(B,F,N,M)<<endl;
    cout << " T: " << printT(B,F,N,M);
 return 0;
}
```

(α) Να γράψετε τη συνάρτηση **fill**, η οποία διαβάζει τα στοιχεία του πλέγματος και τα καταχωρίζει στον δισδιάστατο πίνακα **A**.

Κάθε θέση του πίνακα περιέχει έναν από τους ακόλουθους χαρακτήρες:

- **O**: στεγνό χωράφι
- **X**: πλημμυρισμένο χωράφι (το πολύ ένα σε κάθε γραμμή)
- **#**: ανάχωμα (το πολύ ένα σε κάθε γραμμή)

(Μονάδες 2)

(β) Να γράψετε τη συνάρτηση **calc** η οποία υπολογίζει και καταχωρίζει στον παράλληλο μονοδιάστατο πίνακα **B** τη θέση του αναχώματος κάθε γραμμής (δηλαδή τον δείκτη της στήλης μέσα στην οποία βρίσκεται). Επίσης, υπολογίζει και καταχωρίζει στον παράλληλο μονοδιάστατο πίνακα **F** τη θέση του χωραφιού που έχει πλημμυρίσει. Αν σε μια γραμμή δεν υπάρχει ανάχωμα καταχωρίζεται στον πίνακα B η τιμή -1. Αν σε μια γραμμή δεν υπάρχει πλημμυρισμένο χωράφι να καταχωρίζεται στον πίνακα F η τιμή -1.

(Μονάδες 3)

(γ) Να γράψετε τη συνάρτηση **dry** η οποία υπολογίζει και επιστρέφει το πλήθος των χωραφιών **(D)** που θα παραμείνουν στεγνά μετά την πλήρη εξάπλωση της πλημμύρας. Το νερό εξαπλώνεται μόνο οριζόντια μέσα στην ίδια γραμμή. Αν ένα χωράφι είναι πλημμυρισμένο, τότε όλα τα χωράφια στο ίδιο συνεχόμενο οριζόντιο τμήμα της γραμμής πλημμυρίζουν, εκτός αν παρεμβάλλεται ανάχωμα. Δηλαδή, η ροή του νερού σταματά όταν συναντήσει ανάχωμα. Τα αναχώματα δεν υπολογίζονται ως χωράφια.

(Μονάδες 5)

(δ) Να γράψετε τη συνάρτηση **printT** που επιστρέφει τον μέγιστο αριθμό χωραφιών **(T)** που μπορούν να παραμείνουν στεγνά, **ενώ προηγουμένως θα πλημμύριζαν**, αν τοποθετηθούν στις κατάλληλες θέσεις επιπρόσθετα αναχώματα, στο **αρχικό πλέγμα**. Τα αναχώματα δεν τοποθετούνται στη θέση των πλημμυρισμένων χωραφιών. Σε κάθε γραμμή μπορεί να υπάρχει μόνο ένα ανάχωμα. Τα αναχώματα δεν υπολογίζονται ως χωράφια.

(Μονάδες 5)

Παράδειγμα Εισόδου	**Παράδειγμα Εξόδου**	**Επεξήγηση**
8 7 O X O O O O O O O O O O X O O O O O O O O O O O O O O X O O O O X O # O O X O O O O O O O # O O O X O # O O O O	D: 17 T: 16	Μετά την εξάπλωση του νερού (ερώτημα γ) τα χωράφια έχουν την εξής μορφή: X X X X X X X X X X X X X X O O O O O O O X X X X X X X X X X X X X # X X X X X X X O O O # O O O X X # O O O O Μετά την προσθήκη των επιπρόσθετων αναχωμάτων στο αρχικό πλέγμα και την εξάπλωση του νερού (ερώτημα δ) τα χωράφια έχουν την εξής μορφή: X X **#** O O O O O O O O **#** X X O O O O O O O O O O O O **#** X X X X X X X # X X X **#** O O O O O O # O O O X X # O O O O

Ανάλυση λύσης

Στο **ερώτημα (α),** η συνάρτηση fill διαβάζει όλους τους χαρακτήρες του πλέγματος και τους αποθηκεύει στον δισδιάστατο πίνακα A.

Στο **ερώτημα (β),** η συνάρτηση calc εξετάζει κάθε γραμμή του πίνακα. Για κάθε γραμμή αρχικοποιεί:

B[i] = -1;

F[i] = -1;

Αν βρει ανάχωμα #, αποθηκεύει στον πίνακα B τη στήλη όπου βρίσκεται. Αν βρει πλημμυρισμένο χωράφι X, αποθηκεύει στον πίνακα F τη στήλη όπου βρίσκεται.

Στο **ερώτημα (γ),** η συνάρτηση dry υπολογίζει πόσα χωράφια μένουν στεγνά μετά την εξάπλωση της πλημμύρας.

Για κάθε γραμμή:

- Αν δεν υπάρχει πλημμύρα, τότε όλα τα χωράφια μένουν στεγνά.
- Αν υπάρχει ανάχωμα, αυτό δεν μετρά ως χωράφι.
- Αν υπάρχει πλημμύρα και δεν υπάρχει ανάχωμα, τότε πλημμυρίζει όλη η γραμμή.
- Αν υπάρχει πλημμύρα και ανάχωμα, τότε το ανάχωμα σταματά τη ροή του νερού. Τα στεγνά χωράφια είναι αυτά που βρίσκονται στην αντίθετη πλευρά του αναχώματος.

Στο **ερώτημα (δ),** η συνάρτηση printT εξετάζει μόνο τις γραμμές που έχουν πλημμύρα και δεν έχουν ήδη ανάχωμα. Σε κάθε τέτοια γραμμή μπορεί να τοποθετηθεί ένα νέο ανάχωμα είτε αριστερά είτε δεξιά από το πλημμυρισμένο χωράφι.

Υπολογίζονται δύο πιθανά κέρδη:

left = F[i] - 1;

δηλαδή πόσα χωράφια σώζονται αν το ανάχωμα τοποθετηθεί αριστερά από το X.

right = M - F[i] - 2;

δηλαδή πόσα χωράφια σώζονται αν το ανάχωμα τοποθετηθεί δεξιά από το X.

Επιλέγεται το μεγαλύτερο από τα δύο, γιατί ζητείται ο μέγιστος αριθμός χωραφιών που μπορούν να παραμείνουν στεγνά. Το άθροισμα αυτών των μέγιστων κερδών για όλες τις κατάλληλες γραμμές δίνει το T.

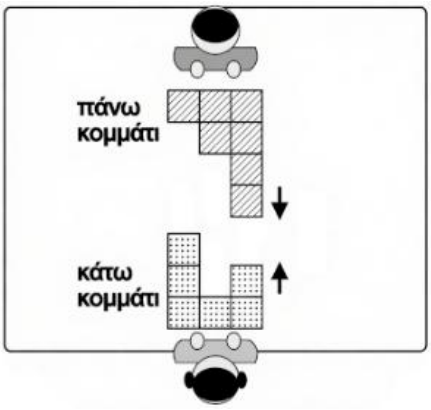
Λύση:

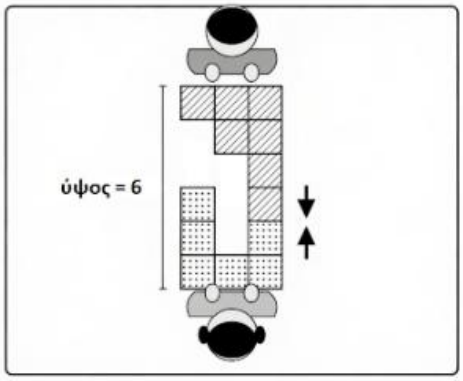
```
void fill(char A[][1000], int N, int M){
    for(int i=0; i<N; i++)
        for(int j=0; j<M; j++)
            cin >> A[i][j];
}



void calc(char A[][1000],int B[],int F[],int N, int M){
    for(int i=0; i<N; i++){
        B[i]=-1;
        F[i]=-1;
        for(int j=0; j<M; j++){
            if(A[i][j]=='#')
                B[i]=j;
            if(A[i][j]=='X')
                F[i]=j;
        }
    }
}

int dry(int B[],int F[],int N,int M){
    int total=0;
    for(int i=0; i<N; i++){
        if(F[i]==-1){
            if(B[i]==-1)
                total+=M;
            else
                total+=M-1;
        }
        else{
            if(B[i]!=-1){
                if(B[i]<F[i])
                    total+=B[i];
                else
                    total+=(M-1)-B[i];
            }
        }
    }
return total;
}
```

```
int printT(int B[], int F[], int N, int M) {
    int best, left, right, total=0;
    for(int i=0; i<N; i++){
        if(F[i]!=-1 && B[i]==-1){
            left=F[i]-1;
            right=M-F[i]-2;
            if(left<0)
                left=0;
            if(right<0)
                right=0;
            if(left>right)
                best=left;
            else
                best=right;
            total+=best;
        }
    }
return total;
}
```

ΑΣΚΗΣΗ 12:

Δύο παιδιά συνεργάζονται για να φτιάξουν ένα παζλ **N** στηλών αποτελούμενο από δύο (2) κομμάτια. Κάθονται απέναντι σε ένα τραπέζι και το κάθε κομμάτι ακουμπάει την άκρη του τραπεζιού (βλέπε εικόνες). Μόλις τελειώσει κάθε παιδί το δικό του κομμάτι, σπρώχνουν και τα δύο κομμάτια μέχρι αυτά να **ακουμπήσουν μεταξύ τους τουλάχιστον σε μια στήλη**.





Η πιο κάτω είσοδος δεδομένων αντιστοιχεί στην πιο πάνω εικόνα:

```
3       - αντιστοιχεί στις 3 στήλες του παζλ
1 2 4   - αντιστοιχεί στα τετράγωνα που έχει το πάνω κομμάτι ανά στήλη
3 1 2   - αντιστοιχεί στα τετράγωνα που έχει το κάτω κομμάτι ανά στήλη
```

Να γράψετε πρόγραμμα, στη γλώσσα προγραμματισμού C++, το οποίο να:

(α) διαβάζει έναν ακέραιο αριθμό **N (2≤N≤16)**, ο οποίος δηλώνει το πλήθος των στηλών του παζλ. Στη συνέχεια, να καταχωρίζει στον μονοδιάστατο πίνακα **top** το πλήθος των τετραγώνων για κάθε μια από τις **N** στήλες του πάνω κομματιού

και ακολούθως, στον μονοδιάστατο πίνακα **bot** το πλήθος των τετραγώνων για κάθε μια από τις **N** στήλες του κάτω κομματιού.

(Μονάδες 2)

(β) υπολογίζει και εμφανίζει το μέγιστο ύψος (**H**) του παζλ.

(Μονάδες 4)

(γ) υπολογίζει και εμφανίζει το συνολικό πλήθος (**E**) των άδειων σημείων ενδιάμεσα των δύο κομματιών.

(Μονάδες 4)

(δ) υπολογίζει και εμφανίζει το πλήθος των κενών σημείων (**F**) από τα οποία αποτελείται η αριστερότερη κενή περιοχή (δύο κενά σημεία ανήκουν στην ίδια περιοχή αν έχουν κοινή πλευρά).

(Μονάδες 5)

Παράδειγμα Εισόδου	**Παράδειγμα Εξόδου**	**Επεξήγηση**
3 1 2 4 3 1 2	H:6 E:5 F:5	T T T . T T . . T B . T B . B B B B
2 1 1 1 1	H:2 E:0 F:0	T T B B
6 1 2 1 1 5 1 1 3 1 2 1 3	H:6 E:14 F:12	T T T T T T . T . . T . . . . . T . . B . . T B . B . B T B B B B B B B
4 5 1 2 3 2 5 2 1	H:7 E:7 F:1	T T T T T . T T T B . T T B . . T B . . B B B . B B B B

Σημείωση: Με τον χαρακτήρα ‘Τ’ εμφανίζονται τα τετράγωνα του πάνω κομματιού, με τον χαρακτήρα ‘Β’ εμφανίζονται τα τετράγωνα του κάτω κομματιού και με τον χαρακτήρα ‘.’ τα κενά σημεία. Η αριστερότερη κενή περιοχή εμφανίζεται με περίγραμμα.

Ανάλυση Λύσης

Ερώτημα β

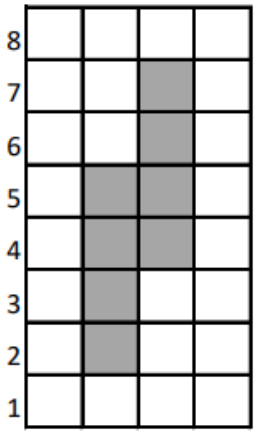
- Το τελικό ύψος του παζλ καθορίζεται από τη στήλη στην οποία τα δύο κομμάτια ακουμπούν μεταξύ τους. Για κάθε στήλη υπολογίζουμε το άθροισμα: top[i] + bot[i]
- Το μέγιστο από αυτά τα αθροίσματα είναι το συνολικό ύψος H του παζλ.

Ερώτημα γ

- Σε κάθε στήλη, τα κενά τετράγωνα είναι όσα απομένουν ανάμεσα στο πάνω και στο κάτω κομμάτι. Άρα για κάθε στήλη τα κενά είναι: H - top[i] - bot[i]
- Το συνολικό πλήθος των κενών σημείων E προκύπτει αθροίζοντας αυτή την ποσότητα για όλες τις στήλες.

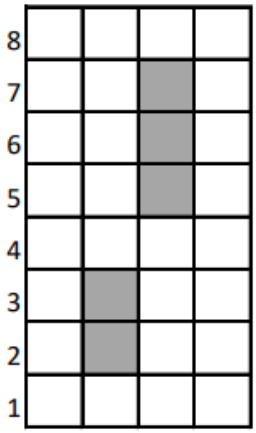
Ερώτημα δ

- Δύο κενές λωρίδες ανήκουν στην ίδια περιοχή όταν έχουν μια κοινή γραμμή. Οι δύο λωρίδες έχουν κοινή γραμμή όταν ισχύουν και τα δύο ταυτόχρονα:
  (1) low(υφιστάμενης) ≤ high(επόμενης): Η υφιστάμενη λωρίδα δεν αρχίζει πιο πάνω από το τέλος της επόμενης.
  (2) low(επόμενης) ≤ high(υφιστάμενης): Η επόμενη λωρίδα δεν αρχίζει πιο πάνω από το τέλος της υφιστάμενης.



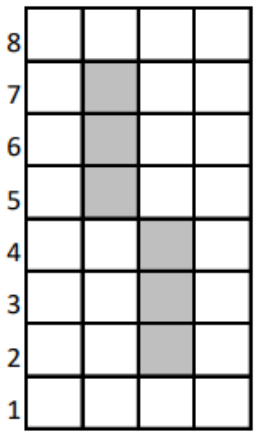
low[c]<=high[c+1] -> true

low[c+1]<=high[c] -> true



low[c]<=high[c+1] -> true

low[c+1]<=high[c] -> false



low[c]<=high[c+1] -> false

low[c+1]<=high[c] -> true

Λύση:

```
#include <iostream>
using namespace std;

int main(){
    int N, H=0, E=0, F=0, c, top[16], bot[16];
// A
    cin >> N;
    for(c=0; c<N; c++)
      cin >> top[c];
    for(c=0; c<N; c++)
      cin >> bot[c];

// B
    for(c=0; c<N; c++)
        if(top[c]+bot[c]>H)
            H=top[c]+bot[c];
// Γ
    for(c=0; c<N; c++)
        E+=H-top[c]-bot[c];
// Δ
    c=0;
    while(c<N && top[c]+bot[c]==H)
        c++;                          // προσπέρασμα γεμάτων στηλών
    if(c<N) {
        F=H-top[c]-bot[c];    // άνοιγμα της 1ης κενής περιοχής
        while (c+1<N
           && bot[c+1]+top[c+1]<H    // η επόμενη στήλη έχει κενά
           && bot[c+1]+1<=H-top[c]   // και υπάρχει επικάλυψη
           && bot[c]+1<=H-top[c+1]){
             c++;
             F+=H-top[c]-bot[c];     // επέκταση της περιοχής
        }
    }

    cout << "H:" << H << " E:" << E << " F:" << F;
 return 0;
}
```

- ΤΕΛΟΣ Γ΄ ΜΕΡΟΥΣ -

- ΤΕΛΟΣ ΕΞΕΤΑΣΗΣ -