21ος ΠΑΝΕΛΛΗΝΙΟΣ ΔΙΑΓΩΝΙΣΜΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΘΕΜΑ Β΄ ΦΑΣΗΣ

(Μαθητές Λυκείου, ΕΠΑΛ, ΕΠΑΣ)

ΧΑΛΚΙΔΙΚΟ ΑΛΦΑΒΗΤΟ

ΕΝΔΕΙΚΤΙΚΕΣ ΛΥΣΕΙΣ

Οι παρακάτω λύσεις είναι απολύτως ενδεικτικές.

C

Αρσένης Γεράσιμος
2ο ΓΕΛ Μοσχάτου

```
/*
LANG: C
TASK: evripos
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define MAX 205

typedef struct {
   int x, y;
} coords;
coords point[MAX];

double a[MAX][MAX];
int n;

double dist(int i, int j) { return sqrt( (point[i].x-
point[j].x)*(point[i].x-point[j].x) + (point[i].y-
point[j].y)*(point[i].y-point[j].y) ); }

int cmp(const coords *i, const coords *j) { return i->x - j->x; }

int main() {
   FILE *fin=fopen("evripos.in", "r"), *fout=fopen("evripos.out",
"w");
   int i, j, k, x, y;
   double min, tmp;

   fscanf(fin, "%d", &n);
   n+=2;
   point[1].x = point[1].y = 0;
```

```
    for (i=2; i<=n; i++) {
       fscanf(fin, "%d %d", &x, &y);
       point[i].x=x;
       point[i].y=y;
    }

    qsort(point+1, n, sizeof(coords), cmp);

    a[1][2] = dist(1, 2);

    for (j=2; j<=n; j++)
       for (i=1; i<=j-1; i++) {
          if (i==1 && j==2) a[i][j] = dist(i, j);
        else if (i<j-1)
          a[i][j] = a[i][j-1] + dist(j, j-1);
        else {
          a[i][j] = a[1][i] + dist(1, j);
          for (k=2; k<=i-1; k++) {
             tmp = a[k][i]+dist(k, j);
             if (tmp < a[i][j]) a[i][j] = tmp;
          }
       }
       }

    min = a[n-1][n] + dist(n-1, n);
    fprintf(fout, "%d\n", (int)round(min));
    return 0;
}
```

C++

Γαϊτανίδης Απόστολος
Ιδ. ΓΕΛ «Μαντουλίδη»

```
/*
LANG: C++
TASK: evripos
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#define inf 0x3f3f3f3f
#define MAXN 210
typedef struct{
   int x,y;
}point;
double dist(point a,point b){
```

```
    return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
double min(double a,double b){
    if(a<b)return a;
    else return b;
}
int cmp(const void *a,const void *b){
    point *c = (point *)a,*d = (point *)b;
    if(c->x >d->x)return 1;
    else return -1;
}
int n;
point p[MAXN];
double a[MAXN][MAXN];
int main(){
    FILE *fin = fopen("evripos.in","r"),*fout = fopen("evripos.out","w");
    fscanf(fin,"%d",&n);
    p[1].x = p[1].y = 0;
    n+=2;
    for(int i=2;i<=n;i++){
        fscanf(fin,"%d %d",&p[i].x,&p[i].y);
    }
    qsort(p+1, n, sizeof(point), cmp);
    a[1][2] = dist(p[1], p[2]);
    for(int j=2;j<=n;j++){
        for(int i =1;i<=j-1;i++){
            if(i==1 && j==2)a[i][j] = dist(p[i],p[j]);
            else if(j>i+1)a[i][j] = a[i][j-1] + dist(p[j-1],p[j]);
            else {
                a[i][j] = a[1][i] + dist(p[1],p[j]);
                for(int k=1;k<=i-1;k++){
                    double q= a[k][i] + dist(p[k],p[j]);
                    if(q<a[i][j]){
                        a[i][j] = q;
                    }
                }
            }
        }
    }
    double m = a[n-1][n] + dist(p[n-1],p[n]);
    for (int i=1; i<n; i++) { m = min(m, a[i][n] + dist(p[i],p[n])); }
    fprintf(fout,"%d\n",(int)round(m));
return 0;
```

}

PASCAL

Καρύδης Θρασύβουλος
ΓΕΛ Κέρκυρας

```
PROGRAM evripos;
  {LANG: Pascal
   TASK: evripos}
  USES
     SysUtils;
  CONST
      MaxPoints = 204;
      INF =    100000;
  TYPE
      coordinates = record
            X:integer;
            Y:integer;
      end;
      Coordsys = array[1..MaxPoints] of coordinates;
  VAR
    N,
    I,J,K,
    d:integer;
    q:real;
    input,output:text;
    points:Coordsys;
    l:array[1..MaxPoints,1..MaxPoints] of real;
  FUNCTION dist(a,b:coordinates):real;   //euclidean distiance//
     BEGIN
       dist:=sqrt(sqr(a.X-b.X)+sqr(a.Y-b.Y))
     END;
  PROCEDURE swap(VAR a, b: coordinates);
     VAR
        t: integer;
     BEGIN
        t   := a.X;
        a.X := b.X;
        b.X := t;
        t   := a.Y;
```

```
      a.Y := b.Y;
      b.Y := t;
   END;
FUNCTION Split(start, stop: integer): integer;
   VAR
      left, right: integer;
      pivot: integer;
   BEGIN
       pivot := points[start].X;
       left := start + 1;
       right := stop;
       WHILE left <= right DO BEGIN
           WHILE (left <= stop) AND (points[left].X < pivot) DO
               left := left + 1;
           WHILE (right > start) AND (points[right].X >= pivot) DO
               right := right - 1;
           IF left < right THEN
             swap(points[left], points[right]);
           END;
       swap(points[start], points[right]);
       Split := right
   END;
PROCEDURE QuicksortRecur(start, stop: integer);
       VAR
          splitpt: integer;
       BEGIN
          IF start < stop THEN BEGIN
             splitpt := Split(start, stop);
             QuicksortRecur(start, splitpt-1);
             QuicksortRecur(splitpt+1, stop);
          END
       END;
PROCEDURE Quicksort(size: Integer; VAR points: Coordsys);
    BEGIN
       QuicksortRecur(1, size)
    END;
BEGIN
   assign(input,'evripos.in');
   reset(input);
   readln(input,N);
   N:=N+2;
   points[1].X:=0;
   points[1].Y:=0;
```

```
    readln(input,points[N].X,points[N].Y);
    FOR I := 2 TO N-1 DO
      readln(input,points[I].X,points[I].Y);
    close(input);
    Quicksort(N, points);
    FOR J := 2 TO N DO
       FOR I := 1 TO J-1 DO
        IF (I=1) AND (J=2) THEN
          l[I,J] := dist(points[i],points[j])
        ELSE IF J>I+1 THEN
              l[I,J]:= l[I,J-1] + dist(points[J-1],points[J])
            ELSE  BEGIN
                l[I,J] := INF;
                FOR K := 1 TO I-1 DO BEGIN
                  q:= l[K,I] + dist(points[K],points[J]);
                  IF q < l[I,J] THEN
                    l[I,J]:= q;
                  END;
                END;
    d:= Round( l[n-1,n] + dist(points[n-1],points[n]));
    assign(output,'evripos.out');
    rewrite(output);
    writeln(output,d);
    close(output);
end.
```

Τα σχόλια παραλείφθηκαν