NEPTUN kód:	NÉV: Aláírás:

feladat	pont	min	elért
1.	10	–	
2.	10	–	
3.	10	–	
4.	16	–	
5.	14	–	
∑	60	21	

Programozás. NZH, 2019. december 9. – BME-TTK, fizika BSc

Arcképes igazolvány hiányában nem kezdheted meg a ZH-t. A feladatok megoldására összesen 90 perc áll rendelkezésre. Az 1. feladatot ezen a lapon oldd meg. A feladatlapot névvel ellátva akkor is be kell adni, ha semmilyen megoldást sem készítettél. Minden feladatmegoldást **külön lapra** írj! Minden lapra írd fel a neved és a neptun kódod, valamint a feladat sorszámát! Ezek hiányában a feladatot nem értékeljük. Szabványos C megoldásokat értékelünk csak. A kétoldalas C referencia összefoglaló használható, más írott, nyomtatott vagy elektronikus segédanyag használata tilos.

..20 – elégtelen, 21..30 – elégséges, 31..40 – közepes, 41..50 – jó, 51.. – jeles

---

1. feladat (5×2 p)

a) Definiálj egy **típust** (*Pont*) polárkoordinátájukkal adott síkbeli pontok tárolására.

```
typedef struct pont{double r,phi;}Pont;
```

b) Adott *n* egy egész típusú változó, ami egy háromjegyű pozitív értékkel bír. Add meg azt a **kifejezést**, mely az *n* számjegyeinek összegét szolgáltatja! Pl. 923 → 14.

```
n%10 + n%100/10 + n/100
```

c) Add meg azt a bitenkénti operátoros **kifejezést**, mely az `unsigned char k` változó legmagasabb, 7. bitjét 1-re *állítja*, az összes többit változatlanul hagyja. (A legkisebb helyiértékű a nulladik bit.)

```
k |= 1u<<7
```

d) Írj **függvényt**, mely – megfelelő hívás esetén – képes a hívó paraméterként átadott valós változójának értékét 0-ra állítani.

```
void zero(double *a){
    *a=0;
}
```

e) Írj **kiíró függvényt**, amely átvesz egy `Pont` tömböt (lásd 1.a.), és az elemeket a szabványos kimenetre írja!

```
void pont_kiir(Pont *t, int n){
    int i;
    for(i=0;i<n;++i)
        printf("%f %f\t", t[i].r, t[i].phi);
}
```

2. feladat 10 p

Írj egy olyan C **függvényt**, amely paraméterként kap két karaktertömböt, valamint egy karaktert és az első tömbben lévő stringet átmásolja a másodikba, miközben minden másolt karakter mögé beszúrja a paraméterként kapott karaktert. Pl. `s1:"abcde"`, `c:'X'`, az eredmény `s2:"aXbXcXdXeX"`.

Add meg a tesztelő pár soros **kódrészletét**, mely létrehoz egy inicializált stringet, és egy másikat az eredmény számára, majd a fenti függvény segítségével előállítja az eredmény szöveget.

```
#include <stdio.h>
void fesul(char *s1, char *s2, char c)
{
    while(*s1!=0)
    {
        *s2++=*s1++;
        *s2++=c;
    }
    *s2='\0';
}
int main()
```

```
{
   char s1[6]="abcde", s2[11];
   fesul(s1,s2,'X');
   printf("%s",s2);
   return 0;
}
```

3. feladat 10 p

A gépünkön "tönkrementek" a szabványos C könyvtárban a math.h függvényei. Taylor azonban megsúgta, hogy $\cosh(x) = \frac{x^0}{0!} + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!}...$ Írj **függvényt**, amely a paraméterlistán kap két valós számot (x, ε), kiszámolja, és visszaadja $\cosh(x)$ közelítő értékét oly módon, hogy a fenti sor újabb tagját figyelembe véve a közelítő érték már ε-nál kevesebbet változik. (Sajnos a `fabs, pow, log, exp` stb. függvények is hibásak.)

```
double mych(double x, double eps)
{
   double xn=1, fakt=1, res, res_e;
   int n=0;
   res_e=xn/fakt;
   n+=2;
   xn=x*x;
   fakt=2;
   res=res_e+xn/fakt;
   while(res-res_e>eps)
   {
      res_e=res;
      n+=2;
      xn*=x*x;
      fakt*=n*(n-1);
      res+=xn/fakt;
   }
   return res;
}
```

4. feladat 16 p

Írj egy **programot**, mely a szabványos bemenetről (billentyűzet) polárkoordinátákkal adott síkbeli pontokat olvas be file vége jelig, de legfeljebb 50 darabot (használd az 1.a. típusát). A program határozza meg, és írja ki az egyes pontokba mutató helyvektorok hosszának a *mediánját*. A medián a hosszúságok szerint sorbarendezett tömb középső eleme (páros elemszám esetén a két középső közül bármelyik választható).

Írj **függvényt**, mely a fent leírtak szerint rendezi a paraméterként átvett pontokból álló tömböt,
VAGY
Írj a könyvtári `qsort` számára megfelelő paraméterezésű és viselkedésű **hasonlító függvényt**.

Írj **függvényt**, mely meghatározza a paraméterként átvett pontokból álló tömbben a helyvektor hosszúságok átlagát, majd egy *épp megfelelő* méretű dinamikus tömböt hoz létre az *átlagnál kisebb* hosszal rendelkező pontok számára, és beleteszi az ilyen pontokat. Gondoskodj az új tömb és elemszáma alkalmas módon történő visszaadásáról.

Írd meg a **programot**, mely beolvassa a pontokat, sorbarendezi azokat, végül kiírja a mediánt, valamint az átlagosnál rövidebb helyvektorú pontokat (hívhatod az 1.e. függvényét). *Ha foglaltál dinamikus memóriát, ne felejtsd el azt felszabadítani.*

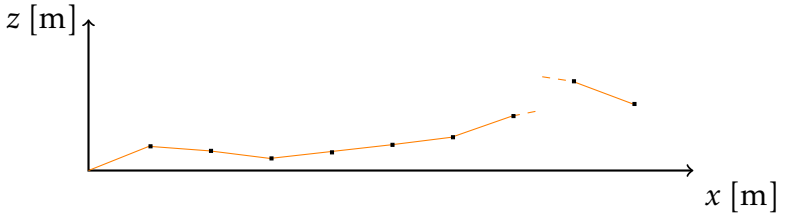
```
#include <stdio.h>
#include <stdlib.h>
```

```c
/*1a*/
typedef struct pont{double r,phi;}Pont;

/*1e*/
void pont_kiir(Pont *t, int n){
   int i;
   for(i=0;i<n;++i)
      printf("%f %f\t", t[i].r, t[i].phi);
}

int hasonhossz(const void *a, const void *b)
{
   Pont *A=(Pont*)a, *B=(Pont*)b;
   if(A->r < B->r) return -1;
   else if(A->r > B->r) return 1;
   else return 0;
}

double atlaghossz(Pont *p, int n)
{
   double s=0;
   for(int i=0; i<n; ++i)
      s+=p[i].r;
   return s/n;
}

Pont *valogat(Pont *p, int n, int *n2)
{
   double atl=atlaghossz(p,n);
   int db=0;
   for(int i=0;i<n;++i)
      if(p[i].r<atl)
         ++db;
   Pont *uj=(Pont*)malloc(db*sizeof(Pont));
   for(int i=0,j=0;i<n;++i)
      if(p[i].r<atl)
         uj[j++]=p[i];
   *n2=db;
   return uj;
}

int main()
{
   Pont p[50],*val;
   int n=0,n2;
   while(n<50 && scanf("%lf %lf",&p[n].r,&p[n].phi)==2)
      ++n;
   qsort(p,n,sizeof(Pont),hasonhossz);
   printf("\nMedian: %f %f\n",p[n/2].r,p[n/2].phi);
   val=valogat(p,n,&n2);
   pont_kiir(val,n2);
   free(val);
   return 0;
}
```

`szamol<szoveg.be>szoveg.db`

5. feladat 14 p

A Pilisben túrázunk. Terv szerint egyenesen kelünk át a hegyeken és völgyeken, jobbra-balra nem kanyarog az utunk.



Egy térkép adatbázisból segítséget is kapunk a tervezéshez, utunk vetületének minden méteréhez (x) megkapjuk a tengerszint feletti magasság értékét (z), valós számként, méterben mérve (lásd a grafikont). Ezt a valós z számsorozatot kapja a programunk a szabványos bemenetén (mintha valaki begépelné), az utolsó érvényes adat után egy 0 érték érkezik, ez jelzi az adatsor végét.

Írj egy olyan C nyelvű teljes **programot**, amely meghatározza és a szabványos kimenetre írja a szabványos bemenetről olvasott adatok ismeretében:

a) a túra során mekkora a vetületen mérve *leghosszabb ereszkedő szakasz*,
b) mekkora a *legnagyobb szintemelkedésű monoton szakasz* a vetületen mérve,
c) van-e, és ha igen, mely x koordinátáknál a tervezett egyenes útvonal mentén *szakadék*? (Ha egy métert haladva x szerint z ennél többet változik.)

A túra hosszára nézve *semmilyen* feltételezéssel nem élhetünk. Ellenben azt feltételezheted, hogy legalább 2 adat érkezik, az első kettő adat különböző, és nem egy szintben mozgunk végig. *Ha foglaltál dinamikus memóriát, ne felejtsd el azt felszabadítani.*

```
#include<stdio.h>
#include<math.h>
#include<stdbool.h>
int main()
{
   bool emelkedo;
   double x=1, z, zr,
         a_maxh=0, a_h=1,
         b_maxh=0, b_h=1;
   scanf("%lf%lf",&zr,&z);
   emelkedo=z>zr;
   while (z!=0)
   {
      if(!emelkedo && z<=zr)
         a_h+=1;
      if(emelkedo && z>=zr)
         b_h+=1;
      else if (emelkedo && z<zr)
      {
         emelkedo=false;
         if(b_h>b_maxh)
            b_maxh=b_h;
         b_h=1;
      }
      else if(!emelkedo && z>zr)
      {
         emelkedo=true;
         if(a_h>a_maxh)
            a_maxh=a_h;
         a_h=1;
      }
      if(fabs(z-zr)>1)
```

```
            printf("szakadek %f es %f meter kozott\n",x-1,x);
        zr=z;
        scanf("%lf",&z);
        x+=1;
    }
    if(emelkedo && b_h>b_maxh) b_maxh=b_h;
    if(!emelkedo && a_h>a_maxh) a_maxh=a_h;
    printf("Leghosszabb ereszkedo: %fm\n",a_maxh);
    printf("Legnagyobb szintemelkedes hossza:%fm\n",b_maxh);
    return 0;
}
```