1. feladat (max. 5×2 pont). Minden szükséges definíciót (pl. ciklusváltozó) le kell írni!

a) Adott *c* egy karakter típusú változó. Adja meg azt a **kifejezést**, amely akkor logikai igaz, ha *c* egy *hexadecimális* számjegyet tartalmaz! (0..9, a..f, A..F)

```
('0'<=c && c<='9') ||
('a'<=c && c<='f') ||
('A'<=c && c<='F')
```

b) Írjon **függvényt**, mely — megfelelő hívás esetén — képes a main függvény két valós változójának értékét kicserélni.

```
void csere(double *a, double *b)
{
   double tmp=*a;
   *a=*b; *b=tmp;
}
```

c) Adottak az alábbi definíciók:

```
char s[]="...", *p, *t, c='x';
```

Írjon **kódrészletet**, mely megkeresi *c* karakter *utolsó* előfordulását az *s* stringben, és ráállítja a *p* pointert. Ha a karakter nem fordul elő a stringben, *p* értéke nullpointer legyen.

pointerekkel:

```
p=NULL;
t=s;
while(*t)
{
   if(*t==c)
      p=t;
   ++t;
}
```

indexeléssel:

```
int i=0;
p=NULL;
while(s[i])
{
   if(s[i]==c)
      p=&s[i];
   ++i;
}
```

d) **Mit ír ki** az alábbi kódrészlet?

```
int a=10,b=2;
printf("%d %d", a<<b, a^b);
```

```
40 8
```

e) Írjon **függvényt**, amely logikai igaz értéket szolgáltat, ha a paraméterként kapott egész szám osztóinak száma *páratlan*, egyébként hamisat!

```
int paratlanosztok(int n){
   int t=2, sz=2; /*1 es n biztosan*/
   while(t<=n/2){
      if(n%t==0)
         ++sz;
      ++t;
   }
   return sz%2;
}
```

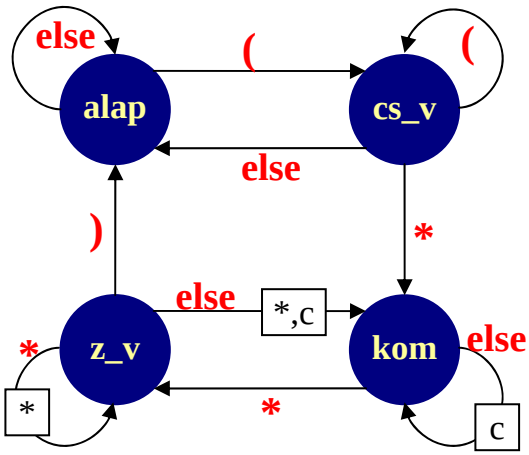
"okos" algoritmus: csak négyzetszámnak van páratlan számú osztója:

```
int paratlanosztok(int n){
   int gy=(int)sqrt(n);
   return gy*gy==n;
}
```

2. feladat (max. 10 pont)

Írjon egy olyan C nyelvű teljes **programot**, amely a szabványos bemenetről file végéig olvasott szintaktikailag helyes Pascal programból *kinyeri* a megjegyzések tartalmát, és kiírja a szabványos kimenetre. Pascalban a megjegyzéseket a `(* *)` karaktersorozatok határolják. Adjuk meg a probléma állapotgráfját is (5 pont). (Feltételezhetjük, hogy a `(*` karakter-szekvencia nem fordul elő szövegkonstansokban.)

```
#include <stdio.h>
typedef enum {alap,cs_v,kom,z_v} allapot;
int main(){
   allapot a=alap; int c;
   while((c=getchar())!=EOF)
   {
      switch(a){
      case alap:
         if(c=='(') a=cs_v;
         break;
      case cs_v:
         if(c=='*') a=kom;
         else if(c!='(') a=alap;
         break;
      case kom:
         if(c=='*') a=z_v; else putchar(c);
         break;
      case z_v:
         if(c=='*') putchar(c);
         else if(c==')') a=alap;
         else {putchar('*');putchar(c);a=kom;}
         break;
      }
   }
   return 0;
}
```



3. feladat (max. 10 pont)

A gépünkön "tönkrementek" a szabványos C könyvtárban a trigonometrikus függvények. Taylor azonban megsúgta, hogy $\sin(x) = +\frac{x^1}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \ldots$

Írjon olyan szabványos C **függvényt**, amely a paraméterlistán kap egy valós számot (x), kiszámolja, és visszaadja sin(x) közelítő értékét a fenti sor első 8 tagját figyelembe véve.

```
double mysin(double x)
{
   double pm=1, xn=x, f=1, res=0;
   int n=1;
   while(n<2*8)
   {
      res += pm*xn/f;
      pm=-pm;
      xn*=x*x;
      n+=2;
      f*=n*(n-1);
   }
   return res;
}
```

---

5. feladat (max. 16 pont)

A Bakonyban túráztunk, Pénzesgyőrből indultunk, és ugyanoda értünk vissza. Utunk során a pillanatnyi helyzetünket egy GPS egység percenként rögzítette. A túra után a GPS egységből az alábbi formátumban kapjuk meg a descartes-i koordinátarendszerbe konvertált adatokat: előre nem ismert (gyakorlatilag a túra hossza percben, plusz 1) darab valós számhármas, az egyes pontok koordinátái. Az utolsó érvényes adat után egy (–1.0; –1.0; –1.0) hármast kapunk.
Ezt a GPS-ből származó adatsorozatot kapja majd a programunk a szabványos bemenetén.

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

a) hány perces, és milyen szintkülönbségű a túra *leghosszabb idejű emelkedője* (monoton növekvő *z* koordinátájú, szakasz) (tipp: egy állapotváltozó hasznos lehet)

b) mekkora volt a legmeredekebb emelkedő szakasz meredeksége ($dz / \sqrt{dx^2+dy^2}$ arány maximuma) ?

A túra hosszára nézve *semmilyen* feltételezéssel nem élhetünk, az csak a GPS adatokból derül ki. Ellenben azt feltételezheti, hogy legalább 2 adat érkezik, és nem egy szintben mozogtunk végig.
Ha foglalt dinamikus memóriát, ne felejtse el azt felszabadítani.

```
#include<stdio.h>
#include<math.h>
int main()
{
   double x,y,z,xx,yy,zz, z1,z1_max, m, m_max;
   int t,tmax=0, fel=0;
   scanf("%lf%lf%lf",&xx,&yy,&zz);
   scanf("%lf%lf%lf",&x,&y,&z);
   m_max=(z-zz)/sqrt((x-xx)*(x-xx)+(y-yy)*(y-yy));
   while(x+1 || y+1 || z+1)
   {
      if(!fel && z>=zz) {fel=1;z1=zz;t=1;}
      else if(fel && z>=zz) {++t;}
      else /*fel && z<zz*/ {fel=0;if(t>tmax) {z1_max=zz-z1;tmax=t;}}
      m=(z-zz)/sqrt((x-xx)*(x-xx)+(y-yy)*(y-yy));
      if(m>m_max)
          m_max=z;
      xx=x;yy=y;zz=z;
      scanf("%lf%lf%lf",&x,&y,&z);
   }
   if(fel && t>tmax) {z1_max=zz-z1;tmax=t;}
   printf ("A leghosszabb emelkedes %d perc, a szint %fm, max meredekseg %f\n",
             tmax,z1_max,m_max);
   return 0;
}
```

4. feladat (max. 14 pont)

Definiáljon egy alkalmas **adatszerkezetet** zenei CD-k alábbi adatainak logikusan rendszerezett tárolására:
Cím (max. 60 karakter), előadó (max. 40 karakter), megjelenés éve (egész) és percben mért játékidő (valós). Rendelje hozzá a `cd` típusazonosítót az így létrehozott új típushoz.

Írjon egy olyan szabványos C **függvényt**, amely a szabványos bemenetről olvasott adatok alapján létrehoz és adatokkal feltölt egy cd-k adatainak tárolására alkalmas tömböt. Ügyeljen arra, hogy a függvény két értéket kell visszaadjon a hívónak. A szabványos bemeneten az alábbi formában kapjuk az adatokat, soronként egy albumot (az előadó és a cím nem tartalmaz szóközt). Például:

```
371     (a lemezek száma az adatbázisban)
1978 36.97 Kate_Bush Lionheart
1975 44.18 Pink_Floyd Wish_You_Were_Here
...     (most épp összesen 371 album adatai egy-egy sorban)
```

Írjon egy olyan szabványos C **programot**, amely a fent megírt függvény segítségével beolvassa a CD-k adatait, egy *már létező* függvény hívásával (lásd alább) a *megjelenésük éve* szerint *növekvő* sorrendbe rendezi a tömb elemeit, majd a rendezett listából az 1973 és 1982 *között kiadottakat* a következő minta szerint, lemezenként két sorba kiírja a szabványos kimenetre:

```
Wish_You_Were_Here /44.18 perc/
Pink_Floyd (1975)
```

Ha foglalt dinamikus memóriát, ne felejtse el azt felszabadítani.
Az alábbi függvényt *NEM KELL* megírnia, de felhasználhatja: `void rendez(cd*,int);` — A paraméterként kapott tömb elemeit az előírt sorrendbe rendezi.

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

typedef struct{
      char cim[60+1], eloado[40+1];
      int ev;
      double ido;} cd;

void rendez(cd*,int);

cd *olvas(int *db)
{
      cd *t; int i;
      scanf("%d",db);
      t=(cd*)malloc(*db*sizeof(cd));
      for(i=0;i<*db;++i)
            scanf("%d %lf %s %s",&t[i].ev,&t[i].ido,t[i].eloado,t[i].cim);
      return t;
}

int main()
{
      int n,i=0;
      cd *t;
      t=olvas(&n);
      rendez(t,n);
      while(i<n && t[i].ev<1973)
            ++i;
      while(i<n && t[i].ev<=1982)
      {
            printf("%s /%.2f perc/\n%s (%d)\n",t[i].cim,t[i].ido,
                                          t[i].eloado,t[i].ev);
            ++i;
      }
      free(t);
      return 0;
}
```