A 2017/2018 tanévi Országos Középiskolai Tanulmányi Verseny első fordulójának feladatai

INFORMATIKA II. (programozás) kategória

1. feladat: Repülők (20 pont)

Ismerünk városok közötti repülőjáratokat. Ha két város között nincs közvetlen járat, akkor is el lehet jutni bármelyikből bármelyikbe valahány átszállással. Ha egy város repülőterét köd miatt lezárják, akkor természetesen sem oda nem jöhetnek repülők, sem onnan nem indulhatnak. Add meg az alábbi járatok alapján, hogy mely városok azok, amelyek közül egyikük repülőterét lezárva a többi városra nem teljesül, hogy bárhonnan bárhova eljuthatunk valahány repüléssel!

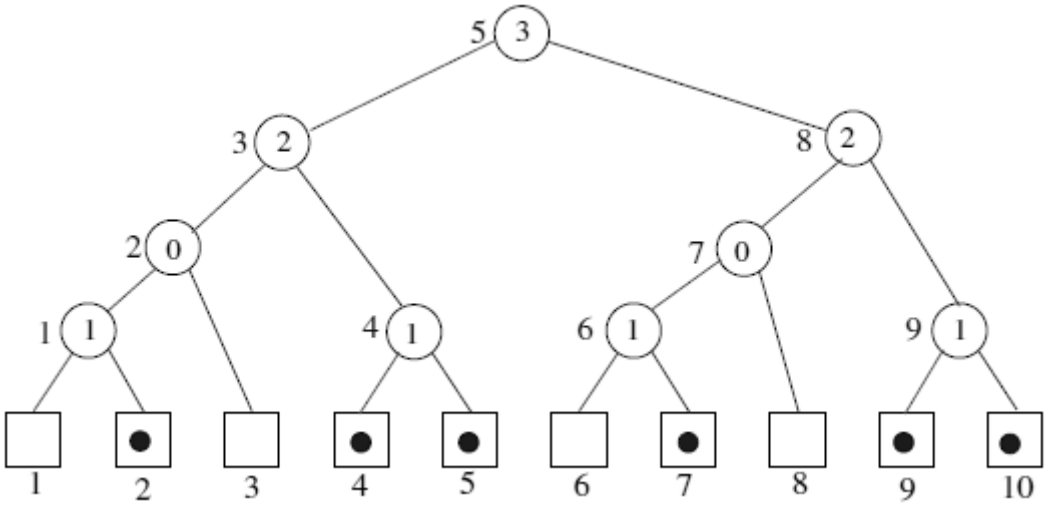
Az A és B város közötti oda-vissza járatokat (A,B) számpár jelöli.

A. (5,2), (1,3), (1,4), (2,3), (4,3)

B. (9,5), (6,5), (10,6), (10,9), (3,2), (3,10), (8,3), (1,8), (7,8), (4,1), (1,10), (1,7), (2,10)

2. feladat: Halmaz (34 pont)

Az {1,…,N} halmaz egy részhalmazát két vektorral ábrázoljuk, a D vektor tartalmazza az ábrán körrel jelölt elemeket, a H vektor pedig a négyzettel jelölteket. H[i] igaz értékű (azaz az ábrán van benne egy pötty), ha az i érték benne van a H-val jelölt halmazban, egyébként hamis.



A. Fogalmazd meg általánosan, mi van a D vektor elemeiben!

B. Az alábbi függvény valami(3,1,10) hívásra milyen értéket ad eredményül és hogyan változtatja a D és a H vektort?

C. Ha ezután a valami(4,1,10) hívással folytatjuk, akkor mi lesz a függvény értéke és hogyan változik a D és a H vektor?

D. Fogalmazd meg általánosan, hogy tetszőleges, a fenti szerkezetű D és H vektorokra mi lesz a függvényérték és hogyan változik a két vektor!

E. Mi a feltétele, hogy a valami(i,1,10) hívásra helyes eredményt kapjunk?

F. Mi történne a fenti vektorokra, ha a valami(10,1,10) hívást hajtanánk végre?

```
valami(k,bal,jobb):
  Ha bal=jobb akkor
    H[bal]:=hamis; valami:=bal
  különben
    s=(bal+jobb)/2
    Ha D[s]≥k akkor D[s]:=D[s]-1; valami:=valami(k,s+1,jobb)
    különben valami:=valami(k-D[s],bal,s)
  Elágazások vége
Függvény vége.
```

3. feladat: Mit csinál? (30 pont)

Az alábbi algoritmus M darab, az X vektorban tárolt 1 és 2^N-1 közötti értékű egész számmal dolgozik. (Az algoritmusban `div` az egész osztást, `mod` a maradékképzést jelenti.)

```
Algoritmus(N,M,X):
  k:=1
  Ciklus i=1-től N-ig
    db:=0;
    Ciklus j=1-től M-ig
      Ha (X[j] div k) mod 2=0 akkor db:=db+1; X[db]:=X[j]
      különben Y[j-db]:=X[j]
    Ciklus vége
    Ciklus j=db+1-től M-ig
      X[j]:=Y[j-db]
    Ciklus vége
    k:=k*2                                              {*}
  Ciklus vége
Eljárás vége.
```

A. Mi lesz az X vektor értéke a {*}-gal jelölt pontnál a külső ciklus egyes lépései után, ha kezdetben N=3, M=7, X=(3,7,5,2,6,1,4)

B. Fogalmazd meg általánosan, hogy mi a feladata az eljárásnak!

C. Fogalmazd meg általánosan, hogy ezt milyen módon teszi!

4. feladat: Összefésüléses rendezés (28 pont)

Az összefésüléses rendezés algoritmusa a következő elven működik:

- az egyelemű sorozat rendezett, nincs vele tennivaló;
- ha a sorozat több elemű, akkor
  - középen két részre osztjuk;
  - mindkét részt rendezzük az összefésüléses rendezés algoritmusával;
  - végül a két kapott rendezett sorozatot összefésüljük.

Az alábbi algoritmus ezt csinálná (az X tömb. E. és U. eleme közötti részt rendezné), azonban sajnos hibák kerültek bele. Jelöld, melyek a hibák!

```
Rendez(E,U):
  Ha E<U akkor K:=(E+U)/2
               Rendez(E,K); Rendez(K,U)
               Összefésül(E,U,K)
  Elágazás vége
Eljárás vége.

Összefésül(E,K,U):
  i:=E; j:=K+1; D:=E; Y:=X
  Ciklus amíg i≤K és j≤U
    D:=D+1
    Ha Y[i]<Y[j] akkor  X[D]:=Y[j]; i:=i+1
               különben X[D]:=Y[j]; j:=j-1
  Ciklus vége
  Ciklus amíg i≤U
    D:=D+1; X[D]:=Y[i]; i:=i+1
  Ciklus vége
  Ciklus amíg j<U
    D:=D+1; X[D]:=Y[j]; j:=j+1
  Ciklus vége
Eljárás vége.
```

5. feladat: Sűrű részsorozat (26 pont)

Adott egy N karaktert tartalmazó S szó és M ($1 \le M \le N$) egész szám. Válasszunk ki az S szó $1 \le i_1 < i_2 < \ldots < i_k \le N$ pozícióit úgy, hogy bármely M hosszú összefüggő [j,j+M-1] intervallum tartalmazzon legalább egy kiválasztott pozíciót! A kiválasztott pozíciókban lévő betűkből képezhető lexikografikusan (ábécé szerint) legkisebb szó legyen R! Az a cél, hogy úgy válasszuk ki a pozíciókat, hogy R a lehető legkisebb legyen a lexikografikus rendezés szerint. Az ilyen R szót kell megadni!

Példa:

M=3, S="cdabc" → R="a"

M=2, S="abcab" → R="aab", azaz megoldás az 1-2-4 pozíción levő betűk sorba rendezve. További lehetőségek (1-3-4, 1-3-5, 2-4, 2-3-5) betűi sorba rendezve ennél ábécében későbbi szót adnak: aac, abc, ab, abc.

A. M=2, S="aaba"

B. M=3, S="aabbba"

C. M=3, S="bacdeabba"

D. M=3, S="bacadebabbda"

E. M=3, S="bdbababdcbbebbabbfbb"

F. M=3, S="axxxbaabadacadeaa"

6. feladat: Üzenetek (29 pont)

Egy hálózaton üzeneteket szeretnénk küldeni. Minden üzenetet egy n hosszú sorozatként kódolunk, ahol a sorozat elemei az 1, …, q számok. (Pl.: n=3, q = 4-re [2, 1, 4] vagy [1, 1, 1] üzenetek). A hálózatunk nem megbízható, néha az átvitel során megváltozik a sorozat néhány tagja. Legyen két üzenet távolsága azon pozícióknak a száma, ahol a két sorozat különbözik. Jelöljük u és v távolságát d(u, v)-vel. (Pl.: d([2, 1, 4], [1, 1, 1]) = 2, mert az első és a harmadik pozícióban tér el a két üzenet.) A hibák javítása érdekében nem használjuk az összes lehetséges üzenetet, csak azok egy C

részhalmazát, ezt nevezzük kódnak. Ha egy kódban nem található sorozatot fogadunk, azt lecseréljük a hozzá legközelebbi kódbeli sorozatra, ezzel megkísérelve a hiba javítását. Egy C kód távolságán a $d(C)=\min_{u,v\in C,\ u\neq v} d(u,v)$ értéket értjük. Egy kód t-hibafelismerő, hogy ha legfeljebb t hiba bekövetkezése esetén felismerjük, hogy az átvitel során hiba történt. Egy kód pontosan t-hibafelismerő, ha t-hibafelismerő, de nem (t+1)-hibafelismerő. Hasonlóan egy kód t-hibajavító, ha legfeljebb t hiba bekövetkezése esetén a sorozatot helyesen javítani tudjuk. Egy kód pontosan t-hibajavító, ha t-hibajavító, de nem (t+1)-hibajavító.

Legyen n=5, q=4, C={[4,2,3,4,1],[1,4,3,4,3],[1,3,1,1,2],[2,2,2,3,4]}!

A. d([4, 2, 3, 4, 1], [1, 4, 3, 4, 3])=?

B. d(C) = ?

C. Pontosan hány hibafelismerő a C kód?

D. Pontosan hány hibajavító a C kód?

E. Mennyi egy T kód d(T) távolsága, ha az pontosan t-hibafelismerő?

F. Mennyi lehet egy T kód d(T) távolsága, ha az pontosan t-hibajavító?

G. Legfeljebb hány elemű lehet egy T kód, ha n = 5, q = 3 és T 2-hibajavító? Adj meg egy ilyen kódot úgy, hogy az egyik sorozat a kódban a [2, 3, 1, 2, 3] legyen!

7. feladat: A Rek bolygó programozói (33 pont)

Nemrég rendezték a programozók intergalaktikus találkozóját, ahol a földi programozók találkoztak a Rek bolygóról érkező kollégáikkal. A földiek meg akarták mutatni a kedvenc programjaikat, de kiderült, hogy a Rek bolygón nem ismerik a ciklusokat és az értékadást. Nincs is rá szükségük, mert mindent rekurzióval valósítanak meg (vagyis olyan függvényeket írnak, amik meghívják saját magukat). A függvények értéke náluk egyszerűen az utoljára kiszámolt kifejezés értéke lesz. Szem előtt tartják a hatékonyságot is, ezért csak végrekurziót használnak. Ez azt jelenti, hogy rekurzív függvényhívás csak az adott függvényben végrehajtott utolsó utasítás lehet. A lenti példák közül az első nem végrekurzív, mert a faktoriális(N-1) kiszámítása után még végre kell hajtani egy szorzást, a második példa viszont már végrekurzív.

Egy programot már sikerült átírnia a tolmácsnak úgy, hogy az idegenek is értsék, de a többiben néhány helyen bizonytalan.

Földi program	Rek program

faktoriális(N):
  a ≔ 1
  Ciklus i=1-től N-ig
  a ≔ a * i
  Ciklus vége
  faktoriális ≔ a
Függvény vége.
	faktoriális(N):
	  ha N = 1 akkor
	  1
	  különben
	  N * faktoriális(N-1)
	Függvény vége.

```
faktoriális(N):
  a ≔ 1
  Ciklus i=1-től N-ig
    a ≔ a * i
  Ciklus vége
  faktoriális ≔ a
Függvény vége.
```

```
faktoriális(N):
  segéd(N,1)
Függvény vége.

segéd(N,M):
  ha N = 0 akkor
    M
  különben
    segéd(N-1,N*M)
Függvény vége.
```

Írd be a hiányzó kifejezéseket! Szerencsére már csak függvény paramétereket, számokat és alapműveleteket kell használnod, mást ne írj be! A programok paraméterei mind pozitív egészek lehetnek.

```
valami(A,B):
  Ciklus amíg A ≠ B
    ha A > B akkor
      A ≔ A - B
    különben
      B ≔ B - A
  Ciklus vége
  valami ≔ A
Függvény vége.
```

```
valami(A,B):
  ha [      ] = [      ] akkor
    A
  különben ha [      ] > [      ] akkor
    valami([      ],[      ])
  különben
    valami(A,[      ])
Függvény vége.
```

```
valami(A,B):
  c ≔ 1
  Ciklus amíg B > 0
    ha B mod 2=1 akkor
      c := c * A
    A := A * A
    B ≔ B / 2
  Ciklus vége
  valami ≔ c
Függvény vége.
```

```
valami(A,B):
  valami2(A,B,[      ])
Függvény vége.

valami2(A,B,C):
  ha [      ] = [      ] akkor
    [      ]
  különben ha [      ] mod 2 = 1 akkor
    valami2([      ],[      ],[      ])
  különben
    valami2([      ],[      ],[      ])
Függvény vége.
```

```
valami(A):
  s ≔ 0
  ciklus i=1-tól A-ig
    b ≔ min(i,A-i+1)
    ha b mod 2=1 akkor
      s ≔ s + b * b
    különben
      s ≔ s - 2 * b
  Ciklus vége
  valami ≔ s
Függvény vége.
```

```
valami(A):
  valami2(□,□,1,□)
Függvény vége.

valami2(A,B,C,D):
  ha B > A akkor
    D
  különben ha □ = □ akkor
    □
  különben
    valami3(□,□,□,□)
Függvény vége.

valami3(A,B,C,D):
  ha B > A akkor
    D
  különben ha □ = □ akkor
    □
  különben
    valami2(A-1,□,□,□)
Függvény vége.
```

Összpontszám: 200 pont