Teză la informatică - clasa a XI-a
Semestrul I – 17.12.2018

NR.1

1. Considerăm subprogramul f definit alăturat. Ce valoare are f(7,11)? Dar f(11,7)? **(1P)**

```
int f(int x,int y)
{ if(x<=y) return x-y;
  return f(y-x,x-1)+3;
}
```

2. Utilizăm metoda backtracking pentru generarea tuturor modalităţilor de a scrie numărul 9 ca sumă a cel puţin două numere naturale nenule distincte. Termenii fiecărei sume sunt în ordine strict crescătoare. Soluţiile se generează în ordinea: 1+2+6, 1+3+5, 1+8, 2+3+4, 2+7, 3+6 şi 4+5. Se aplică exact aceeaşi metodă pentru scrierea lui 8. Câte soluţii vor fi generate? **(1P)**

3. Scrieți definiția completă a **subprogramului recursiv** *P* care primește prin intermediul parametrului *n* un număr natural nenul (*n≤100*), iar prin intermediul parametrului *x* un tablou unidimensional cu *n* componente întregi, de maximum opt cifre fiecare. Subprogramul furnizează prin intermediul parametrului *mini* valoarea minimă din tabloul *x*, prin intermediul parametrului *maxi* valoarea maximă din *x*, iar prin intermediul parametrului *sum* suma elementelor din tabloul x. **(2P)**

   - *0 < n <= 100*
   - numele subprogramului cerut este *P*
   - parametrii sunt, în această ordine: *x*, *n*, *mini*, *maxi*, *sum*
   - elementele vectorului *x* sunt indexate de a zero

4. La un festival sunt programate *n* spectacole, pentru fiecare se cunoaşte momentul de început și momentul de sfârșit, exprimate prin numere naturale. Un spectator dorește să urmărească cât mai multe spectacole în întregime. Determinați numărul maxim de spectacole care pot fi urmărite, fără ca acestea să se suprapună. Fişierul de intrare *spectacole.in* conţine pe prima linie numărul *n*. Pe fiecare dintre următoarele *n* linii se află câte două numere naturale *X Y*, reprezentând momentul de început și momentul de sfârșit al unui spectacol. Fişierul de ieşire *spectacole.out* va conţine pe prima linie numărul *S*, reprezentând numărul maxim de spectacole care pot fi urmărite, fără să se suprapună. *1 ≤ n ≤ 100;* momentele de început și sfârșit ale spectacolelor sunt numere naturale mai mici decât 1000000 ; pentru fiecare spectacol, *X < Y ;* dacă momentul de început al unui spectacol și momentul de sfârșit al altui spectacol coincid, pot fi urmărite ambele spectacole. **(2.5P)**

5. Se citesc două numere naturale nenule *n* și *m*. Să se determine toate şirurile cu *m* elemente din mulţimea *{1,2,..,n}*, ordonate strict crescător, cu proprietatea că oricare două elemente consecutive în şir au diferenţa mai mică sau egală cu cu *2*. Fişierul de intrare *siruri.in* conţine pe prima linie numerele *n* şi *m*, separate printr-un spaţiu. Fişierul de ieşire *siruri.out* va conţine pe fiecare linie câte *m* valori, separate prin câte un spaţiu, reprezentând elementele unui şir. *1 ≤ m ≤ n ≤ 15;* şirurile vor fi afişate în ordine lexicografică .
   (2.5P)

Teză la informatică - clasa a XI-a
Semestrul I – 17.12.2018

NR.2

1. Pentru definiţia alăturată a subprogramului **f**, ce valoare are **f(3) şi f(8)**? **(1P)**

```
int f(int x)
{ if(x<=4) return x*x-3;
return f(x-3)+4;
}
```

2. Utilizăm metoda backtracking pentru generarea tuturor modalităţilor de a scrie numărul **9** ca sumă a cel puţin două numere naturale nenule distincte. Termenii fiecărei sume sunt în ordine strict crescătoare. Soluţiile se generează în ordinea: **1+2+6**, **1+3+5**, **1+8**, **2+3+4**, **2+7**, **3+6** şi **4+5.** Se aplică exact aceeaşi metodă pentru scrierea lui **12**. Scrieţi, în ordinea generării, toate soluţiile de forma **2+...** **(1P)**

3. Se dă un sir cu n elemente, numere naturale nenule. Folosind metoda Divide et Impera, determinaţi cel mai mare divizor comun al elementelor acestui şir. Programul citește de la tastatură numărul n, iar cele n elemente ale șirului. Programul afișează pe ecran numărul X, reprezentând cel mai mare divizor comun al elementelor. Restricţii şi precizări $1 \leq n \leq 1000$; elementele şirului vor avea cel mult 9 cifre ;
se recomandă folosirea metodei Divide et Impera. **(2P)**

4. În curtea unui atelier de reparaţii auto, sunt n maşini care trebuie sa fie reparate. Deoarece nu sunt suficienţi mecanici, în fiecare moment de timp se poate lucra doar la o singură maşină. Cunoscând timpul necesar pentru repararea fiecărei maşini, scrieţi un program care calculează numărul maxim de maşini care pot fi reparate într-un interval de timp T. Pe prima linie a fişierului masini.in se găsesc două numere naturale n şi T separate printr-un singur spaţiu, reprezentând numărul de maşini din curtea atelierului auto şi timpul total în care se va lucra. Pe linia a doua, separate prin câte un spaţiu, se găsesc n numere naturale t1, t2, …, tn, reprezentând timpii necesari pentru repararea fiecărei maşini. Pe prima linie a fişierului masini.out se va găsi un număr natural k, reprezentând numărul maxim de maşini care pot fi reparate. $1 < n$, $T <= 1000$; numerele de pe a doua linie a fişierului de intrare vor fi mai mici sau egale cu 100. **(2.5P)**

5. Fie mulţimea M={1,2,..,n} şi P(1),P(2),...,P(n) o permutare a ei. Elementul x din M se numeşte punct fix dacă P(x)=x. Se citeşte un număr natural nenul n. Să se afişeze, în ordine lexicografică, permutările fără puncte fixe ale mulţimii {1,2,..,n}. Fişierul de intrare permpf.in conţine pe prima linie numărul n. Fişierul de ieşire permpf.out va conţine pe fiecare linie elementele unei permutări, separate prin câte un spaţiu. **(2.5P)**