

Testare finală - Grupa de excelență CNILC1 semigrupa 2 An școlar 2014-2015

Partea I (20p)

1. Se consideră următorul șir, construit astfel încât fiecare element al lui, cu excepția primului, se obține din cel precedent: 1, 11, 21, 1211, 111221, .. Scrieți care este al nouălea termen din șir. (10p)
2. Completați programul pentru a verifica în mod corect dacă două cuvinte sunt anagrame. (10p)

```
1 #include<iostream>
2 #include<cstring >
3 char x[2000], y[2000], *p;
4 int main(){
5     cin>>x>>y;
6     if (strlen(x)==strlen(y)) {
7         while (strchr(y,x[0])!=NULL && x[0]!=0){
8
9             .....
10
11         }
12     if (x[0]==0 && y[0]==0) cout<<"Anagrama";
13     else cout<<"Nu sunt anagrama";
14 }
15 else cout<<"Nu sunt anagrama";
16 }
```

Partea II (70p)

1. Realizați o secvență de instrucțiuni care determină toate numerele prime mai mici decât numărul natural N ($N < 1000000$), folosindu-se de algoritmul numit „Ciurul lui Eratostene”. Complexitate cerută: $O(N\sqrt{N})$. (10p)
2. Se consideră două șiruri de N , respectiv M caractere ($N, M \leq 100$). Asupra primului cuvânt se pot face următoarele schimbări în așa fel încât acesta să ajungă identic cu al doilea:
 - se poate șterge o literă;
 - se poate insera o literă;
 - se poate schimba o literă în altă literă.Scrieți secvență de instrucțiuni care construiește matricea C unde elementul $C_i[j]$ memorează minim de schimbări necesare pentru a potrivi primele i valori din primul șir și primele j valori din al doilea șir. Complexitate cerută $O(N*M)$ *Exemplu* Pentru ana și alla se va afișa 2. (10p)
3. Realizați un subprogram care permite restaurarea unui *maxheap* a cărui rădăcină nu respectă condiția de maxim în raport cu fiii săi. Se presupune că ambii subarbori ai rădăcinii sunt *maxheap*-uri. Subprogramul primește prin intermediul parametrului r , indicele rădăcinii iar, prin parametrul k , numărul de elemente din *maxheap*. *Exemplu*: Pentru $H=(5,23,44,18,16,6,10)$, se va afișa $H=(44,23,10,18,16,6,5)$. Complexitate $O(\log(N))$. (20p)
4. Realizați un program care ordonează un șir de N perechi de numere naturale după principiul sortării *RadixSort*. Fișierul de intrare se numește **radixsort.in**, respectiv **radixsort.out**. Se vor folosi doar vectori statici. Complexitate cerută: $O(N)$, spațiu de memorie maxim $O(5*N)$. Toate valorile din fișier sunt mai mici ca 100000 (30p)
 $N=4$ și perechile (4, 5) (1, 3) (1, 5) (4, 3) | Se va afișa (1,3)(1,5)(4,3)(4,5)

Se acordă 10 p din oficiu