Examen – Programarea Calculatoarelor
Seria CD

Data: 5 februarie 2019 **Durata: 90 de minute**

SUBIECT 1 (28 puncte + 2 puncte bonus)

După ce ați absolvit cursul de Programarea Calculatoarelor, aveți de rezolvat în limbajul C o problemă ce presupune modelarea unui joc de șah. Pentru aceasta, aveți următoarele structuri de date:

- Tablă de șah
  - Dimeniune; Număr piese – numere întregi fără semn, maxim un byte (tabla este pătratică)
  - Nume jucători – vector de două șiruri de caractere, maxim 100 de caractere pentru fiecare nume
  - Piese – vector către piesele de pe tablă, alocat dinamic
- Piesă de șah
  - Tip piesă – un byte care trebuie să rețină atât culoarea piesei (alb/negru), cât și tipul său (6 tipuri diferite)
  - Număr mutări – numărul de mutări de până acum făcute de piesă
  - Istoric mutări – un vector alocat dinamic, conține câte două poziții per mutare, cu toate mutările anterioare. Ultima mutare este poziția curentă a piesei.
  - Un pointer către o funcție care întoarce mutările posibile pentru piesa curentă, cu semnătura:
    *unsigned char** pozitiiViitoare(TPiesa* piesa, TTabla* tabla)*

1.1. Să se definească tipurile de date *TTabla* și *TPiesa*. *TTabla* va conține câmpurile: *dim*, *nrPiese*, *nume* și *piese*, iar *TPiesa* va conține câmpurile: *tip*, *numMutari*, *mutari* și *pozViitoare* (4 puncte)

1.2. Să se scrie o funcție care alocă memorie pentru un element de tipul *TPiesa* cu antetul:
TPiesa alocaPiesa(unsigned int numMutari)* (2 puncte)

1.3. Scrieți o funcție care alocă memorie pentru o tabla cu *nrPiese*, cu următoarea semnatură:
TTabla alocaTabla(unsigned char nrPiese, unsigned int *numMutari)* (2 puncte)

1.4. Definți o funcție care citește datele de intrare dintr-un fișier binar deschis deja pentru citire. În fișier, întâi sunt stocate numărul de piese și dimensiunea istoricului de mutări pentru fiecare piesă. După aceea, în fișier se găsesc toate informațiile necesare completării unei table, cu tot cu piesele aferente, mai puțin pointerii către funcții (alegeți voi ordinea în care aceste informații se găsesc în fișier). (6 puncte)

1.5. Implementați funcția *pozitiiViitoareTura* care generează pozițiile următoare pentru o tură. Apoi implementați funcția de mai jos care iterează prin toate piesele și schimbă pointerul către funcția corectă de mutare pentru toate turele de pe tablă:
*void schimbaFunctieMutareTura(TTabla *tabla)* (8 puncte)

1.6. Realizați o funcție care să sorteze crescător toate piesele după suma celor două coordonate ale unei piesei.
TPiese sorteazaPiese(TTabla *tabla)* (5 puncte)

1.7. Definiți o funcție care eliberează toată memoria alocată:
*void dezalocaTabla(TTabla** tabla)* (3 puncte)

SUBIECT 2 (12 puncte = 3 x 4 puncte)

Care este ieșirea (warning-uri sau erori de compilare, erori de execuție, mesaje afișate, comportament nedefinit) secvențelor de cod de mai jos? Justificați pe scurt, în maxim 5 rânduri, răspunsul vostru. Pentru fiecare exercițiu, considerați ca toate headerele standard (stdio.h, stdlib.h, etc.) sunt incluse automat.

2.1

```
#define N 4
int fun (void *p ) {
    char* pc = (char*)p;
    (*(pc + 1))++;
}

int main() {
    int x = 257, i = 0;
    while (i < N) {
        printf("%d\n", x);
        fun(&x);
        i++;
    }
    return 0;
}
```

2.2

```
void fun(char *p) {
    if (p == NULL || *p == '\0') return 0;
    int current = 1, i = 1;
    while (*(p+current)){
        if (p[current] != p[current-1]){
            p[i] = p[current];
            i++;
        }
        current++;
    }
    *(p+i)='\0';
}
int main() {
    char str[] = "annaareeraanna";
    fun(str);     puts(str);
    return 0;
}
```

2.3

```
int fun (int n, int *fp ) {
    int t, f;
    if (n <= 1){
        *fp = 1;
        return 1;
    }
    t = fun(n-1, fp);
    f = t + *fp;
    *fp = t;
    return f;
}
int main() {
    int x = 15;
    printf("%d\n",fun(6, &x));
    printf("%d\n", x);
    return 0;
}
```