Examen – Programarea Calculatoarelor
Seria CD

Data: 31 ianuarie 2020 **Durata: 90 de minute**

SUBIECT 1 (28 puncte + 2 puncte bonus)

Ce ar fi sa îmbinăm un pic Programarea și Matematica?
Haideți să ne jucăm cu niște serii de numere întregi, folosind următoarele structuri:

- Serie
  - Parametrii serie – maxim 4 bytes în care veți stoca tipul seriei (primul byte cu semnificația 0 - serie numere prime, 1 – serie Fibonacci, 2 – serie aritmetică, 3 – serie geometrică, al doilea și al treilea byte sunt folosite doar de progresiile aritmetică și geometrică și semnifică rația, respectiv primul termen)
  - Număr elemente – întreg fără semn, numărul de termeni care trebuie generați din seria curentă
  - Termeni – vector de numere întregi fără semn, cu putere de reprezentare mare, conține termenii seriei
  - Generator – pointer către o funcție care primește parametrii seriei și numărul de elemente și întoarce un șir de numere naturale, primii termeni ai seriei
- ColectieSerii
  - Descriere – șir de caractere care conține descrierea colecției, alocat dinamic
  - Număr serii – întreg fără semn, conține numărul de serii din colecția curentă
  - Serii – vector alocat dinamic ce conține pointeri către variabile de tip Serie

1.1. Să se definească tipurile de date *TSerie* și *TColectie*. *TSerie* va conține câmpurile: *param*, *num, termeni* și *generator*, iar *TColectie* va conține câmpurile: *desc*, *numSerii* și *serii* (3 puncte)

1.2. Să se scrie o funcție care alocă memorie pentru un element de tipul *TSerie*, inițializează funcția generator (puteți numi cele 4 funcții de generare cum doriți) și o apelează pentru a calcula termenii seriei, cu antetul:
TSerie alocaSerie(int num, …)* (4 puncte)

1.3. Scrieți o funcție care alocă memorie pentru o colecție de *numSerii* progresii aritmetice (primul termen 1 și rații consecutive începând cu 1), știind numărul de elemente pentru fiecare din aceste serii, cu următoarea semnatură:
TColectie alocaColectie(unsigned int numSerii, unsigned int *numTerms)* (3 puncte)

1.4. Implementați funcțiile de tip generator care calculează primii termeni pentru șirurile de numere prime, respectiv Fibonacci. (6 puncte)

1.5. Realizați o funcție care completează descrierea unei colecții de șiruri și o scrie într-un fișier text numit “out.txt”. Descrierea conține câte un rând pentru fiecare serie de forma: “Seria 1 de tip progresie aritmetica cu media 5.0” (atenție cum calculați media pentru șiruri cu 0 elemente!). Semnătura funcției este:
*void completeazaSiScrieDescriere(TColectie** colectie)* (7 puncte)

1.6. Implementați o funcție care verifică dacă există un termen care să aparțină tuturor seriilor, cu antetul de mai jos. Dacă nu există, întoarceți -1, iar dacă există mai mulți, pe cel mai mare dintre aceștia.
*unsigned long long termenComun(TColectie** colectie)* (4 puncte)

1.7. Definiți o funcție care eliberează toată memoria alocată pentru o colecție de serii:
*void dezalocaColectie(TColectie** colectie)* (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

```
void f(char *s) {
    printf("%c", *s);
    if (strlen(s) >= 1) {
        f(s + 1);
    }
    printf("%s", s);
}

int main() {
    f("Hello");

    return 0;
}
```

2.2

```
int main() {
    char *x = (char *)calloc(10, sizeof(int));
    for (int i = 0; i < 10; ++i) {
        x[i] = i % 4 == 1;
    }
    for (int i = 0; i < 10; ++i) {
        printf("%d ", ((int *)x)[i]);
    }

    return 0;
}
```

2.3

```
int main() {
    char str[30] = "Trec cu nota mare la PC!";
    memmove(str + 17, str + 12, 13);
    puts(str);
    memcpy(str + 17, str + 12, 13);
    puts(str);

    return 0;
}
```