Examen – Programarea Calculatoarelor și Limbaje de Programare 1
Seria CD

Data: 04 februarie 2022 **Durata: 90 de minute**

SUBIECT 1 (28 puncte + 2 puncte bonus)

După ce ați absolvit cursul PCLP 1, aveți de rezolvat în limbajul C o problemă ce presupune modelarea unor itinerarii de vacanță. Pentru aceasta, aveți următoarele structuri de date:

- Itinerariu
  - Număr trasee – număr întreg fără semn
  - Lungime trasee – vector de numere întregi ce conține dimensiunea fiecărui traseu din itinerariul curent
  - Prima destinație – vector de pointeri către destinații; fiecare pointer va adresa prima destinație de pe traseul curent
- Destinație
  - Nume oraș – șir de maxim 100 de caractere care conține numele orașului vizitat
  - Cod țară – șir de 3 caractere care conține codul ISO al țării din care face parte orașul vizitat
  - Destinația următoare – următoarea destinație de pe traseul curent sau NULL dacă este ultimul oraș

1.1. Să se definească tipurile de date *TItinerariu* și *TDestinatie*. *TItinerariu* va conține câmpurile: *numTrasee*, *lungimeTrasee* și *primaDest*, iar *TDestinatie* va conține câmpurile: *nume*, *tara* și *urmDest* (3 puncte)

1.2. Să se scrie o funcție care alocă memorie pentru un element de tipul *TDestinatie* și copiază orașul și țara primite ca parametri, cu antetul: *TDestinatie* alocaDestinatie(char* oras, char* tara)* (3 puncte)

1.3. Scrieți o funcție care alocă memorie pentru un itinerariu cu *nrTrasee*, știind dimensiunea fiecărui traseu, precum și numele orașelor și țărilor de pe aceste trasee (presupuneți că dimensiunea lor este corectă), cu următoarea semnatură:

TItinerariu alocaItinerariu(unsigned int nrTrasee, unsigned int *lungimeTrasee, char** orase, char** tari)* (4 puncte)

1.4. Definți o funcție care citește datele de intrare dintr-un fișier text deschis deja pentru citire. În fișier, întâi sunt stocate numărul de trasee, apoi lungimea fiecărui traseu, apoi orașele și țările de pe fiecare din aceste trasee (ordine stabilită de către voi). Dacă sunt necesare alte informații, le puteți citi și pe acestea – ordinea țărilor și orașelor o puteți stabili voi. (4 puncte)

1.5. Implementați o funcție care calculează care este orașul care apare cel mai des în toate traseele dintr-un itinerariu. Funcția are un parametru de tip *TItinerariu** și întoarce numele orașului și numărul său de apariții. (8 puncte)

1.6. Realizați o funcție care să întoarcă toate orașele dintr-o țară primită ca parametru, sortate crescător lexicografic (puteți păstra numele de orașe duplicate).

*char** sorteazaOraseDinTara(TItinerariu *itinerariu, char *tara)* (5 puncte)

1.7. Definiți o funcție care eliberează toată memoria alocată: *void dezalocaItinerariu(TItinerariu* itinerariu)* (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 răspunsul vostru – maxim 5 rânduri si eventual un desen. Pentru fiecare exercițiu, considerați ca toate headerele standard (stdio.h, stdlib.h, etc.) sunt incluse automat.

2.1

```
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[] = "aannaareeraannna";
    fun(str);   puts(str);
    return 0;
}
```

2.2

```
#define N 10
void foo(int* x, void* y) {
  printf("%d ", (char*)x - (char*)y);
  ++x;
}
int main(void) {
  int i, *p;
  p = calloc(10, sizeof(int));
  void* p1 = p;
  for (i = 0; i < N; i++, p++){
    foo(p, p1);
  }
  return 0;
}
```

2.3

```
void fun(int n) {
    if(n > 0){
        fun(n-1);
        printf("%d ", n);
        fun(n-1);
    }
}

int main() {
    fun(4);
    return 0;
}
```