Examen – Programarea Calculatoarelor
Seria CD

Data: 1 februarie 2018 **Durata: 90 de minute**

SUBIECT 1 (28 puncte + 2 puncte bonus)

La finalul anului 1, un prieten vă roagă să îl ajutați cu o problemă ușoară de geometrie. Pentru aceasta, sunt necesare următoarele structuri:

- Punct 2D
  - Identificator punct – număr întreg fără semn, maxim un byte
  - Coordonate x și y – două numere double
- Figură geometrică 2D
  - Nume figură – un șir de caractere de maxim 100 caractere
  - Numărul de puncte din figură (pot fi maxim 10)
  - O listă cu punctele care formează figura geometrică curentă
  - Un pointer către o funcție care calculează automat aria figurii geometrice curente, cu semnătura *double funArie(TPunct2D* puncte, int nrPuncte)*

1.1. Să se definească tipurile de date *TPunct2D* și *TFigura2D*. *TPunct2D* va conține câmpurile: *id*, *x* și *y*, iar *TFigura2D* va conține câmpurile: *nume*, *nrPuncte*, *puncte* și *functieArie* (4 puncte)

1.2. Să se scrie o funcție care alocă memorie pentru un element de tipul *TFigura2D* cu antetul:
*TFigura2D *alocaFigura(unsigned int nrPuncte)* (2 puncte)

1.3. Scrieți o funcție care alocă memorie pentru o planșă cu *nrFiguri*, cu următoarea semnatură:
*TFigura2D **alocaPlansa(unsigned int nrFiguri, unsigned int *nrPuncte)*
unde *nrPuncte* reprezintă numărul de puncte asociate fiecărei figuri. (2 puncte)

1.4. Definți o funcție care citește datele de intrare dintr-un fișier binar deschis deja pentru citire. În fișier, întâi este salvat un *unsigned int* care reprezintă numărul de figuri geometrice (*nrFiguri*). După aceea, apar *nrFiguri* numere care reprezintă numărul de puncte din fiecare figură. Apoi veți avea toate informațiile necesare completării unei planșe, atât pentru figuri, cât și pentru punctele asociate (alegeți voi ordinea acestora din fișier). (6 puncte)

1.5. Implementați o funcție care iterează prin toate figurile și asociază funcția care calculează ariile în felul următor: pentru triunghiuri echilaterate numele funcției este *funArieTriEchi*, pentru pătrate numele funcției este *funAriePatrat*, iar pentru restul figurilor se folosește funcția generică *funArieFiguraComplexa*:
*void asociazaFunctieArie(TFigura2D ** plansa, unsigned int nrFiguri)* (8 puncte)

1.6. Realizați o funcție care să sorteze descrescător figurile din planșă în funcție de aria lor.
char sorteazaFiguriArie(TFigura2D ** plansa, unsigned int nrFiguri)* (5 puncte)

1.7. Definiți o funcție care eliberează toată memoria alocată:
*dezalocaPlansa(TFigura2D ** plansa, unsigned int nrFiguri)* (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

```
int fun(int x, int y) {
  if (x == 0)
    return y;
  return fun(x - 1,  x + y);
}
int main(void) {
  int x = 3, y = 6;
  int *p = &x;
  (*p)++;
  printf("%d %d", fun(x, y),
    fun(*p, y));
  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

```
char str1[50];
char *foo(char str[]) {
    static int i = 0;
    if (*str)
    {
        foo(str+1);
        str1[i] = *str;
        i++;
    }
    return str1;
}
int main(void) {
    char str[] = "Examen Programare 2018";
    printf("%s", foo(str));
    return 0;
}
```