Egzamin z Zaawansowanego Programowania Komputerowego

12 czerwca 2023

1. (15 pkt.) Napisać funkcję

```
int a(int* t, int n, int v);
```

która przestawia elementy tablicy `t` o rozmiarze `n` tak, aby wszystkie elementy o wartości mniejszej niż `v` poprzedzały elementy o wartości większej lub równej `v`. Kolejność pomiędzy elementami w poszczególnych grupach może być dowolna. Funkcja powinna zwrócić liczbę elementów o wartości mniejszej niż `v`. Należy zadbać o to, żeby złożoność funkcji była liniowa ze względu na rozmiar tablicy.

2. (15 pkt.) Napisać funkcję

```
int f(int n);
```

która zwraca liczbę przedstawień liczby `n` w postaci sumy różnych liczb pierwszych.
Np. f(3)=1 (**3**=3), f(6)=0, f(7)=2 (**7**=5+2=7), f(18)=4 (**18**=13+5=13+3+2=11+7=11+5+2).

3. (10 pkt.) Dana jest funkcja

```
int c(int a, int b) {
      if(a<0 || b<0 || a*b==0) return a+b;
      return c(a-1, b-1)+a*b;
}
```

Napisać równoważną funkcję, która nie używa rekurencji i podać jej złożoność.

4. (10 pkt.) Dana jest struktura

```
class ElListy { public: int wartosc; ElListy* nast; };
```

której będziemy używać do tworzenia list jednokierunkowych.
Napisać funkcję

```
bool rowne(ElListy* a, ElListy* b);
```

która zwraca true wtedy i tylko wtedy, gdy listy `a` i `b` są równe, tj. mają takie same wartości kolejnych elementów oraz taką samą długość.

5. (15 pkt.) Dana jest struktura

```
class Wezel{ public: int wartosc; Wezel* lewy; Wezel* prawy; };
```

której będziemy używać do tworzenia drzew.
Napisać funkcję

```
int usunLiscie(Wezel* d);
```

która usuwa z drzewa wszystkie liście: węzły, które nie mają poddrzew. Należy zwrócić liczbę usuniętych liści.

6. (15 pkt.) Stworzyć klasę `Miasta`, której obiekty reprezentują miasta oraz sieć dróg pomiędzy nimi. Każda droga łączy dwa miasta i ma podaną długość w km.

```
class Miasta {
public:
      Miasta();
      void dodaj(string m);
      void polacz(string m1, string m2, int odl);
      void usunMiasto(string m);
      void usunPolaczenie(string m1, string m2);
      int odleglosc(string m1, string m2);
      string sasiad(string m);
};
```

Opis metod publicznych, które należy napisać:

`Miasta();` – tworzy pusty zbiór miast

`void dodaj(string m);` – dodaje miasto o podanej nazwie, niepołączone z innymi (jeśli miasto już jest, nic się nie dzieje)

`void polacz(string m1, string m2, int odl);` – dodaje połączenie pomiędzy miastami

`void usunMiasto(string m);` – usuwa miasto oraz wszystkie drogi łączące to miasto z innymi

`void usunPolaczenie(string m1, string m2);` – usuwa połączenie pomiędzy miastami

`int odleglosc(string m1, string m2);` – zwraca długość drogi pomiędzy miastami lub -1 jeśli takiej drogi nie ma (lub jedno z miast nie istnieje)

`string sasiad(string m);` – zwraca nazwę (dowolnego) najbliższego miasta lub `""` jeśli miasto nie jest połączone z żadnym innym