Egzamin z Zaawansowanego Programowania Komputerowego

4 lipca 2022

1. (15 pkt.) Napisać funkcję

```
void g(int** t, int n);
```

która przestawia elementy kwadratowej tablicy `t` o rozmiarze `n` tak, aby wszystkie wiersze i wszystkie kolumny były posortowane rosnąco. Podać złożoność napisanej funkcji.

2. (15 pkt.) Skończony ciąg $(a_1,\ldots,a_n)$ nazwiemy dopuszczalnym jeśli wszystkie jego elementy są całkowite i dodatnie, $a_1=a_n=1$ oraz sąsiednie elementy różnią się co najwyżej o 1 (tj. a_k-a_{k+1} wynosi 0, 1 lub -1 dla wszystkich $0<k<n$). Napisać funkcję

```
bool f(int s);
```

która zwraca liczbę dopuszczalnych ciągów $(a_1,..,a_n)$ takich, że $a_1+..+a_n=s$.
Np. f(6)=5, bo mamy ciągi (1,1,1,1,1,1), (1,2,1,1,1), (1,1,2,1,1), (1,1,1,2,1), (1,2,2,1).

3. (10 pkt.) Dana jest funkcja

```
int c(int n) {
      if(n<=0) return 0;
      if(n==1) return 1;
      if(n%3==0) return c(n/3)+1;
      return c(n+1)+1;
}
```

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

```
ElListy* usunOstatni(ElListy* a);
```

która usuwa ostatni element listy `a`. Funkcja powinna zwrócić wskaźnik do listy (lub 0 jeśli lista po usunięciu jest pusta).

5. (15 pkt.) Dana jest struktura

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

której będziemy używać do tworzenia drzew.
Napisać funkcję

```
bool h(Wezel* d);
```

która zwraca `true` wtedy i tylko wtedy, gdy w każdym węźle suma elementów lewego i prawego poddrzewa tego węzła jest taka sama.

6. (15 pkt.) Stworzyć klasę `Zbior`, której obiekty reprezentują zbiory napisów typu `string`.

```
class Zbior {
public:
      Zbior();
      void dodaj(string s);
      void dodaj(Zbior& z);
      void usun(string s);
      int ile();
      int nalezy(string s);
      Zbior uzupelnienia(string s);
};
```

Opis metod publicznych, które należy napisać:

`Zbior();`	Tworzy pusty zbiór.
`void dodaj(string s);`	Dodaje element `s` do zbioru.
`void dodaj(Zbior& z);`	Dodaje wszystkie elementy ze zbioru `z` do zbioru.
`void usun(string s);`	Usuwa element `s` ze zbioru.
`int ile();`	Zwraca liczbę elementów w zbiorze.
`int nalezy(string s);`	Zwraca `true` jeśli `s` należy do zbioru.

`Zbior uzupelnienia(string s);`
Zwraca nowy zbiór zawierający wszystkie elementy zbioru, których prefiksem jest `s`. Np. jeśli z={"ABC","BAB","ABBC","AAB","C"}, to z.uzupelnienia("AB") powinno zwrócić zbiór {"ABC","ABBC"}.