Egzamin z Zaawansowanego Programowania Komputerowego

7 czerwca 2013

A1. Napisać funkcję

```
int g(int n);
```

która zwraca liczbę przedstawień liczby n w postaci malejącej sumy kwadratów różnych liczb całkowitych dodatnich. Oszacować złożoność napisanej funkcji.
*Przykład: Wywołanie g(5) powinno zwrócić 1 (bo 5=2*2+1*1), a g(25) powinno zwrócić 2 (bo 26=5*5=4*4+3*3).*

Dana jest struktura

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

którą będziemy używać do tworzenia drzew.

A2. Napisać funkcję

```
Wezel* lustro(Wezel* d);
```

która dokonuje odbicia lustrzanego drzewa d, tzn. w każdym węźle lewe i prawe poddrzewo zamieniają się miejscami. Funkcja powinna zwracać wskaźnik do wynikowego drzewa.

A3. Napisać funkcję

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

która zwraca true wtedy i tylko wtedy, gdy drzewo jest uporzadkowane rosnąco, tzn. dla każdego węzła wartości w lewym jego poddrzewie są mniejsze od wartości w węźle, a wartości w prawym poddrzewie są większe.

A4. Napisać klasę Lamana której łamane na płaszczyźnie. Wierzchołki łamanej mają współrzędne rzeczywiste i są ponumerowane od 1 do n. Ponadto zdefiniowana jest klasa

```
class Punkt {
public:
      Punkt(double xx, double yy)  { x=xx; y=yy; }
      double x; double y;
};
```

której obiekty reprezentują wierzchołki łamanej.

```
class Lamana {
public:
      // tworzy łamaną a-b
      Lamana(Punkt a, Punkt b);
      // dodaje punkt p na końcu łamanej
      void dodaj(Punkt p);
      // zwraca ostatni wierzchołek
      Punkt ostatni();
      // usuwa ostatni wierzchołek, jeśli pozostał tylko
      // jeden, nie robi nic
      void usunOstatni();
      // zwraca liczbę wierzchołków
      int ile();
      // pomiędzy każdą parę kolejnych wierzchołków wstawia
      // nowy wierzchołek leżący dokładnie w połowie drogi
      // pomiędzy nimi
      void wstawSrodki();
};
```

Egzamin z Zaawansowanego Programowania Komputerowego

7 czerwca 2013

Dana jest struktura

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

używana do tworzenia drzew.

B1. Napisać funkcję

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

która usuwa z drzewa wszystkie liście, tj. węzły, które nie mają poddrzew. Funkcja powinna zwracać wskaźnik do wynikowego drzewa. Jeśli drzewo zawiera tylko jeden węzeł, należy go skasować a funkcja powinna zwrócić 0.

B2. Napisać funkcję

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

która zwraca true wtedy i tylko wtedy, wszystkie wartości w poddrzewach każdego węzła są mniejsze od wartości w tym węźle.

B3. Napisać funkcję

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

która zwraca liczbę przedstawień liczby n w postaci malejącej sumy różnych liczb całkowitych dodatnich o tej własności, że każdy kolejny składnik jest nie większy niż połowa poprzedniego. Oszacować złożoność napisanej funkcji.

Przykład: f(4) powinno zwrócić 2 (bo 4=4=3+1), a f(8) powinno zwrócić 5 (bo 8=8=7+1=6+2=5+3=5+2+1)

B4. Napisać klasę Ciag, której obiekty reprezentują ciągi liczb rzeczywistych. Wyrazy ciągu są zawsze ponumerowane od 1 do n, po usunięciu wyrazu ze środka należy pozostałe wyrazy w razie potrzeby przenumerować.

```
class Ciag {
public:
      // tworzy nowy jednoelementowy ciąg
      Ciag(double x);
      // zwraca liczbę elementów ciągu
      int ile();
      // dodaje nowy wyraz na koncu ciągu
      void dodaj(double x);
      // usuwa wszystkie wyrazy ujemne, zwraca liczbę
      // usuniętych wyrazów
      double usunUjemne();
      // usuwa ostatni wyraz i go zwraca
      double usunOstatni();
      // drukuje wyrazy ciągu
      void drukuj();
};
```