laboratorium (licencjat wieczorowy): zadanie nr 3 *26 maja 2009 r.*

ALGORYTMY I STRUKTURY DANYCH

zrównoważone drzewo BST

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie:

Zaimplementuj słownik z operacjami słownikowymi *insert*, *delete* i *search* w postaci drzewa SPLAY, AVL albo RBT. W drzewach tych mają być przechowywane liczby całkowite. Klucze w pamiętanym zbiorze dynamicznym nie mogą się powtarzać. Program powinien interpretować i realizować następujące rozkazy:

- Sprawdzenie czy element jest w drzewie (przykładowo 11):
  `search 11`
  W odpowiedzi procedura zwraca wartość 1 gdy element został znaleziony, albo 0 gdy nie było go w drzewie.
- Wstawienie elementu do drzewa (przykładowo 23):
  `insert 23`
  W odpowiedzi procedura zwraca wartość 1 gdy element został wstawiony, albo 0 gdy elementu nie wstawiono (w drzewie był już taki element).
- Usunięcie elementu z drzewa (przykładowo 47):
  `delete 47`
  W odpowiedzi procedura zwraca wartość 1, gdy element został usunięty, albo 0 gdy elementu nie usunięto (w drzewie takiego elementu nie było).
- Sprawdzenie liczby elementów w drzewie:
  `size`
  W odpowiedzi procedura zwraca liczbę wszystkich elementów w drzewie.
- Wypisanie wszystkich elementów zapamiętanych w drzewie w porządku *inorder*:
  `print`
  W odpowiedzi procedura przegląda wszystkie węzły w drzewie w odpowiedniej kolejności i wypisuje zapamiętane w nich wartości w jednej linii (po wypisaniu każdej wartości jest drukowana pojedyncza spacja).

Dane:

W pierwszym wierszu z danymi jest podana liczba n ($1 \leq n \leq 1\,000\,000$) oznaczająca liczbę instrukcji wykonywanych na początkowo pustym zbiorze dynamicznym, a w kolejnych n wierszach zapisane są rozkazy zgodnie ze składnią opisaną wcześniej.

Wyniki:

W wyniku należy wypisać w n wierszach wszystkie n odpowiedzi generowane przez program (każda odpowiedź w osobnym wierszu).

Uwaga:

Wybrane drzewo BST należy zaimplementować samodzielnie, a nie korzystać z gotowych kolekcji w STL'u.

Przykład:

Przykładowe dane wejściowe mogą mieć postać:

```
13
insert 5
search 4
insert 3
delete 2
print
insert 5
insert 2
print
insert 7
delete 2
print
search 5
size
```

Wówczas na wyjścu powinien pojawić się następujący wynik:

```
1
0
1
0
3 5
0
1
2 3 5
1
1
3 5 7
1
3
```