14. Binary search tree

1 Zadanie

1.1 Specyfikacja

Spróbujmy stworzyć podwaliny wirtualnego dziekanatu. Tym razem wykorzystamy strukturę drzewa binarnego. Zaimplementuj program, który umożliwi tworzenie i wstawianie nowych węzłów oraz wyszukiwanie i usuwanie rekordów o podanym numerze indeksu.

Szablon programu należy uzupełnić o definicje następujących funkcji:

1. `node* create_node(const int number, const char* name)`, która alokuje strukturę `node` na stercie, przypisuje odpowiednie wartości jej polom i zwraca jej adres.
2. `void delete_tree(node* root)`, zwalniająca pamięć węzłów drzewa.
3. `node* insert(node* root, node* to_insert)`, wstawiająca węzeł znajdujący sie pod adresem `to_insert` do drzewa. Kluczem jest numer indeksu studenta.
4. `node* find (node* root, const int number)`, znajdująca w drzewie węzeł o danym numerze indeksu. Funkcja zwraca adres znalezionego węzła lub wskaźnik zerowy, gdy żądany węzeł nie został znaleziony.
5. `node* delete(node* root, const int number)`, usuwająca z drzewa węzeł o danym numerze indeksu.

1.2 Wejście

W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite a, r i f; odpowiednio liczba elementów do wstawienia, usunięcia i wyszukania. Kolejne a wierszy zawiera liczbę naturalną i słowo (o długości nie większej niż 31 znaków) - numer indeksu i nazwisko studenta, które należy umieścić w bazie. Następne r wierszy zawiera po jednej liczbie naturalnej - numer indeksu studenta do wykreślenia (niekoniecznie istniejący). Ostatnie f wierszy zawiera po jednej liczbie naturalnej - numer indeksu studenta do wyszukania w systemie.

1.3 Wyjście

Na standardowym wyjściu programu powinno znaleźć się dokładnie f wierszy - nazwisko wyszukiwanego studenta lub napis `NO`, gdy student o szukanym numerze indeksu nie istnieje.

1.4 Przykład

Wejście:

```
5 2 2
101 Nowak
134 Heller
112 Kowalska
142 Pawlak
135 Maliniak
101
144
134
101
```

Wyjście:

```
Heller
NO
```