Zestaw 2C

Ocena:

ocena 3.0	Ocena 4.0	Ocena 5.0
Zad. 1 lub Zad. 2 lub Zad. 3	Zad. 1, Zad. 2 lub Zad. 1, Zad. 3 lub Zad. 2, Zad. 3	Zad. 1, Zad. 2, Zad. 3

Ramówka programu na następnej stronie. Proszę z niej skorzystać. W razie potrzeby proszę odpowiednio zmodyfikować listy argumentów funkcji i typy wartości zwracanych przez funkcje.

Zadanie 1:
Zaimplementuj operacje enqueue i dequeue dla kolejki FIFO przechowującej liczby naturalne. Dowolna reprezentacja kolejki. Należy umożliwić także wyświetlenie stanu kolejki. Implementacja optymalna.

Zadanie 2:
Zaimplementuj operację wyszukiwania minimum w drzewie BST przechowującym liczby całkowite. Wynikiem na być wartość przechowywana w węźle lub -1 jeśli nie istnieje. Reprezentacja tablicowa. Implementacja optymalna.

Zadanie 3:
Zaimplementuj operację dodawania elementu, o podanej wartości, na końcu listy dwukierunkowej oraz operację usuwania elmentu z końca tej listy. Lista przechowuje liczby naturalne. Należy także umożliwić wyświetlenie stanu listy przed i po dodaniu/usunięciu elementu. Implementacja optymalna.

Ramówka program 2C:

```
#include <cstdlib>
#include <iostream>

using namespace std;

//Zadanie 1

void enqueue(int z){}
int dequeue (){}
void print(){} //wydruk stanu kolejki
void zadanie1(){
cout<<”Zadanie 1”<<endl;
enqueue(1); print();enqueue(4); print();enqueue(9); print();enqueue(12); print();enqueue(4); print();
cout<< dequeue();print(); cout<< dequeue();print();
enqueue(5); print();enqueue(10); print();
cout<< dequeue(); print(); cout<< dequeue(); print(); cout<< dequeue(); print();
cout<< dequeue(); print(); cout<< dequeue(); print(); cout<< dequeue(); print(); cout<< dequeue(); print();
cout<<”Zadanie 1 - koniec”<<endl;
}

//Zadanie 2

int minimum(char bst[]){}
void zadanie2(){
cout<<”Zadanie 2”<<endl;
//drzewo BST użyte w zadaniu powinno być wynikiem dodawania kolejno następujących liczb: 9 5 8 12 6 10 2
char bst[] ={};  //należy w odpowiedni sposób zainicjalizować tablicę podanymi znakami
cout<<minimum(bst);
cout<<”Zadanie 2 - koniec”<<endl;
}

//Zadanie 3

void dodaj(int z){}
int usun(){}
void printL(){}
void zadanie3(){
cout<<”Zadanie 3”<<endl;
dodaj(2); dodaj(5); dodaj(4); printL(); dodaj(8); print(); usun(); printL();
usun(); printL();usun(); printL();usun(); printL();usun(); printL();
cout<<”Zadanie 3 - koniec”<<endl;
}

int main(int argc, char *argv[])
{
  zadanie1();
  zadanie2();
  zadanie3();

  system("PAUSE");
  return EXIT_SUCCESS;
}
```