Carlos Rangel
1217700029

Homework 4: Submission Template

1. Copy and paste your complete program for question 1 here.

BinarySearchTree.h:

```
#ifndef BINARYSEARCHTREE_H
#define BINARYSEARCHTREE_H

#include <iostream>
#include <cstdlib>
#include <string>
#include <limits>

class BinarySearchTree{
    private:
        class node{
            public:
                node* left;
                node* right;
                node* parent;
                int key;
                std::string data;
        };
        void destroyTree(node* x){
            if(x != NULL){
                destroyTree(x->left);
                destroyTree(x->right);
                delete x;
            }
        }

    public:
        node* root;
        BinarySearchTree();
        ~BinarySearchTree();
        bool isEmpty();
        BinarySearchTree::node* findMin(node* x);
        void removeMin();
        BinarySearchTree::node* findMax(node* x);
        void removeMax();

        void preOrderTreeWalk(node*);
        void inOrderTreeWalk(node*);
        void postOrderTreeWalk(node*);

        void inOrderTreeWalk_flightName(node*);

        void treeInsert(int);
        void treeInsert(int, std::string);
```

```
        void treeDelete(node* z);
        BinarySearchTree::node* treeSuccessor(node*);
        BinarySearchTree::node* treePredecessor(node*);
        BinarySearchTree::node* treeSearch(node* x, int key);
        void transplant(node* u, node* v);

};
#endif // BINARYSEARCHTREE_H
```

BinarySearchTree.cpp:

```
#include <iostream>
#include <cstdlib>
#include <vector>
#include <limits>

//header file for methods
#include "BinarySearchTree.h"
////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////

//constructor
BinarySearchTree::BinarySearchTree(){
    root = NULL;
}
//destructor
BinarySearchTree::~BinarySearchTree(){
    destroyTree(root);
}

////////////////////////////////////////////////////////////////////////////
//checking da tree and removing min/max
bool BinarySearchTree::isEmpty(){
    return root == NULL;
}

BinarySearchTree::node* BinarySearchTree::findMin(node* x){
    while(x->left != NULL)
        x = x->left;
    return x;
}
BinarySearchTree::node* BinarySearchTree::findMax(node* x){
    while(x->right != NULL)
        x = x->right;
    return x;
}

void BinarySearchTree::removeMin(){
    treeDelete(findMin(root));
}
void BinarySearchTree::removeMax(){
```

Carlos Rangel
1217700029

```
41.     treeDelete(findMax(root));
42. }
43.
44. ////////////////////////////////////////////////////////////////////////////
45. //tree walks
46. void BinarySearchTree::preOrderTreeWalk(node* x){
47.     if(x != NULL){
48.         std::cout << " " << x->key << " ";
49.         if(x->left) inOrderTreeWalk(x->left);
50.         if(x->right)inOrderTreeWalk(x->right);
51.     }
52. }
53. void BinarySearchTree::inOrderTreeWalk(node* x){
54.     if(x != NULL){
55.         if(x->left) inOrderTreeWalk(x->left);
56.         std::cout << " " << x->key << " ";
57.         if(x->right)inOrderTreeWalk(x->right);
58.     }
59. }
60. void BinarySearchTree::postOrderTreeWalk(node* x){
61.     if(x != NULL){
62.         if(x->left) inOrderTreeWalk(x->left);
63.         if(x->right)inOrderTreeWalk(x->right);
64.         std::cout << " " << x->key << " ";
65.     }
66. }
67.
68. //secondary inorder tree walk to be more verbose with flight manager.
69. void BinarySearchTree::inOrderTreeWalk_flightName(node* x){
70.     if(x != NULL){
71.         if(x->left) inOrderTreeWalk_flightName(x->left);
72.         std::cout << " Flight " << x->data << "\t::\tlanding in " << x->key << " minutes.\n";
73.         if(x->right)inOrderTreeWalk_flightName(x->right);
74.     }
75. }
76. ////////////////////////////////////////////////////////////////////////////
77. //insert, delete, and dictionary ops (with helper functions)
78. void BinarySearchTree::treeInsert(int key){
79.     node* z = new node();
80.     z->key = key;
81.     z->left = NULL;
82.     z->right = NULL;
83.     z->parent = NULL;
84.
85.     node* y = NULL;
86.     node* x = root;
87.
88.     while(x != NULL){
89.         y = x;
90.         if(z->key < x->key)
```

Carlos Rangel
1217700029

```
91.             x = x->left;
92.         else x = x->right;
93.     }
94.     z->parent = y;
95.     if(y == NULL)
96.         root = z;
97.     else if(z->key < y->key)
98.         y->left = z;
99.     else y->right = z;
100.}
101.
102.//overload to include string input
103.void BinarySearchTree::treeInsert(int key, std::string data){
104.    node* z = new node();
105.    z->key = key;
106.    z->data = data;
107.    z->left = NULL;
108.    z->right = NULL;
109.    z->parent = NULL;
110.
111.    node* y = NULL;
112.    node* x = root;
113.
114.    while(x != NULL){
115.        y = x;
116.        if(z->key < x->key)
117.            x = x->left;
118.        else x = x->right;
119.    }
120.    z->parent = y;
121.    if(y == NULL)
122.        root = z;
123.    else if(z->key < y->key)
124.        y->left = z;
125.    else y->right = z;
126.}
127.
128.void BinarySearchTree::treeDelete(node* z){
129.    if(z->left == NULL)
130.        transplant(z, z->right);
131.    else if(z->right == NULL)
132.        transplant(z, z->left);
133.    else {
134.        node* y = findMin(z->right);
135.        if(y != z->right){
136.            transplant(y, y->right);
137.            y->right = z->right;
138.            y->right->parent = y;
139.        }
140.        transplant(z, y);
```

Carlos Rangel
1217700029

```
141.        y->left = z->left;
142.        y->left->parent = y;
143.    }
144.}
145.
146.BinarySearchTree::node* BinarySearchTree::treeSuccessor(node* x){
147.    if(x->right != NULL)
148.        return findMin(x->right);
149.    else{
150.        node* y = x->parent;
151.        while(y != NULL && x == y->right){
152.            x = y;
153.            y = y->parent;
154.        }
155.        return y;
156.    }
157.}
158.
159.BinarySearchTree::node* BinarySearchTree::treePredecessor(node* x){
160.    if(x->left != NULL)
161.        return findMax(x->left);
162.    else{
163.        node* y = x->parent;
164.        while(y != NULL && x == y->left){
165.            x = y;
166.            y = y->parent;
167.        }
168.        return y;
169.    }
170.}
171.
172.BinarySearchTree::node* BinarySearchTree::treeSearch(node* x, int key){      //iterative search
173.    while(x != NULL && key != x->key){
174.        if(key < x->key)
175.            x = x->left;
176.        else x = x->right;
177.    }
178.    return x;
179.}
180.
181.void BinarySearchTree::transplant(node* u, node* v){
182.    if(u->parent == NULL)
183.        root = v;
184.    else if(u == u->parent->left)
185.        u->parent->left = v;
186.    else u->parent->right = v;
187.    if(v != NULL)
188.        v->parent = u->parent;
189.}
190.
```

Carlos Rangel
1217700029

```
191.//helper function to validate user input
192.bool checkCin(){
193.    if(std::cin.fail()){
194.        std::cin.clear();
195.        std::cin.ignore(std::numeric_limits<std::streamsize>::min(), '\n');
196.
197.        return false;
198.    }
199.    return true;
200.}
201.////////////////////////////////////////////////////////////////////////////
202.//
203.//COMMENTED OUT FOR LINKING PURPOSES
204.//
205.////////////////////////////////////////////////////////////////////////////
206.///* <= remove to test methods.
207.int main(){
208.    using namespace std;
209.
210.    BinarySearchTree bst;
211.    int choice, key;
212.
213.    while(true){
214.        cout << endl << "\n";
215.        cout <<  "Binary Search Tree Example \n";
216.        cout << " ---------------------------- \n";
217.        cout << " 1.  Insert a Node \n";
218.        cout << " 2.  Delete a Node\n";
219.        cout << " 3.  Search for a key\n";
220.        cout << " 4.  Pre-Order Traversal \n";
221.        cout << " 5.  Post-Order Traversal \n";
222.        cout << " 6.  In-Order Traversal\n";
223.        cout << " 7.  Find Max\n";
224.        cout << " 8.  Remove Max\n";
225.        cout << " 9.  Find Min\n";
226.        cout << " 10. Remove Min\n";
227.        cout << " 11. Successor\n";
228.        cout << " 12. Predecessor\n";
229.        cout << " 13. Exit\n";
230.        cout << " Enter your choice : \n";
231.        cin >> choice;
232.
233.        switch(choice){
234.            case 1:
235.                cout << "\nEnter key to be Inserted (integer value) : ";
236.                cin >> key;
237.                if(!checkCin())
238.                    std::cerr << "Error: invalid input. Please try again...\n" << std::endl;
239.                else{
240.                    bst.treeInsert(key);
```

Carlos Rangel
1217700029

```
241.                     cout << "\n " << key << " was successfully inserted!";
242.                 }
243.                 break;
244.
245.             case 2:
246.                 cout << "\nEnter key to be deleted (integer value) : ";
247.                 cin >> key;
248.                 if(!checkCin())
249.                     std::cerr << "Error: invalid input. Please try again...\n" << std::endl;
250.                 else{
251.                     if(bst.treeSearch(bst.root, key) != NULL){
252.                         bst.treeDelete(bst.treeSearch(bst.root, key));
253.                         cout << "\n " << key << " was successfully deleted!";
254.                     }
255.                     else cout << "\n " << key << " could not be found. . .\n";
256.                 }
257.                 break;
258.
259.             case 3:
260.                 cout << "\nEnter key to search for in the BS-tree : ";
261.                 cin >> key;
262.                 if(!checkCin())
263.                     std::cerr << "Error: invalid input. Please try again...\n" << std::endl;
264.                 else{
265.                     if(bst.treeSearch(bst.root, key) != NULL){
266.                         cout << bst.treeSearch(bst.root, key)->key << " was found!\n";
267.                         if(bst.treeSearch(bst.root, key)->parent != NULL) cout << bst.treeSearch(bst.root, key)->key << "-
    >parent:  " << bst.treeSearch(bst.root, key)->parent->key << "\n";
268.                         if(bst.treeSearch(bst.root, key)->left != NULL)   cout << bst.treeSearch(bst.root, key)->key << "->left: "
    << bst.treeSearch(bst.root, key)->left->key << "\n";
269.                         if(bst.treeSearch(bst.root, key)->right != NULL)  cout << bst.treeSearch(bst.root, key)->key << "->right: "
    << bst.treeSearch(bst.root, key)->right->key << "\n";
270.                     }
271.                     else cout << key << " was not found. . .\n";
272.                 }
273.                 break;
274.
275.             case 4:
276.                 cout << " Pre-Order Traversal \n";
277.                 cout << " --------------------\n";
278.                 bst.preOrderTreeWalk(bst.root);
279.                 break;
280.
281.             case 5:
282.                 cout << "\n Post-Order Traversal \n";
283.                 cout << " --------------------\n";
284.                 bst.postOrderTreeWalk(bst.root);
285.                 break;
286.
287.             case 6:
```

Carlos Rangel
1217700029

```
288.                cout << "\n In-Order Traversal \n";
289.                cout << " --------------------\n";
290.                bst.inOrderTreeWalk(bst.root);
291.                break;
292.
293.            case 7:
294.                cout << "\n Max key found is: ";
295.                cout << bst.findMax(bst.root)->key << "\n";
296.                break;
297.
298.            case 8:
299.                cout << "\n Max key " << bst.findMax(bst.root)->key;
300.                bst.removeMax();
301.                cout << " was removed. . .\n";
302.                break;
303.
304.            case 9:
305.                cout << "\n Min key found is: ";
306.                cout << bst.findMin(bst.root)->key << "\n";
307.                break;
308.
309.            case 10:
310.                cout << "\n Min key " << bst.findMin(bst.root)->key;
311.                bst.removeMin();
312.                cout << " was removed. . .\n";
313.                break;
314.
315.            case 11:
316.                cout << "Please insert key to find the successor of : ";
317.                cin >> key;
318.                if(!checkCin())
319.                    std::cerr << "Error: invalid input. Please try again...\n" << std::endl;
320.                else{
321.                    if(bst.treeSearch(bst.root, key) != NULL){
322.                        cout << "\nThe Predecessor of the node containing " << key << " is ";
323.                        cout << bst.treeSuccessor(bst.treeSearch(bst.root, key))->key << "\n";
324.                    }
325.                    else cout << key << " could not be found. . .\n";
326.                }
327.                break;
328.
329.            case 12:
330.                cout << "Please insert key to find the predecessor of : ";
331.                cin >> key;
332.                if(!checkCin())
333.                    std::cerr << "Error: invalid input. Please try again...\n" << std::endl;
334.                else{
335.                    if(bst.treeSearch(bst.root, key) != NULL){
336.                        cout << "\nThe Predecessor of the node containing " << key << " is ";
337.                        cout << bst.treePredecessor(bst.treeSearch(bst.root, key))->key << "\n";
```

Carlos Rangel
1217700029

```
338.                    }
339.                    else cout << key << " could not be found. . .\n";
340.                }
341.                break;
342.
343.            case 13:
344.                system("pause");
345.                return 0;
346.                break;
347.
348.            default:
349.                cout << "Invalid choice\n";
350.                break;
351.        }
352.    }
353.}
354.//*/
```

Carlos Rangel
1217700029

2. Copy and paste your complete program for question 2 here.

```
1.  ////////////////////////////////////////////////////
2.  // Flight management program by Carlos Rangel
3.  // Made for CSE310 to study and implement BSTree
4.  ////////////////////////////////////////////////////
5.
6.  #include <iostream>
7.  #include <cstdlib>
8.  #include <vector>
9.  #include <limits>
10. #include "BinarySearchTree.h"
11.
12. #define DOTLINE  " ----------------------------------------\n"
13.
14. static bool checkCin(){
15.     if(std::cin.fail()){
16.         std::cin.clear();
17.         std::cin.ignore(std::numeric_limits<std::streamsize>::min(), '\n');
18.
19.         return false;
20.     }
21.     return true;
22. }
23.
24. static void printMenu(){
25.     using namespace std;
26.
27.     cout << DOTLINE;
28.     cout << " A. Request Landing\n";
29.     cout << " B. Withdraw landing request\n";
30.     cout << " C. List landing times & flight Numbers\n";
31.     cout << " ?. Print this menu\n";
32.     cout << " Q. Exit\n\n";
33.
34. }
35.
36. int main(){
37.     using namespace std;
38.
39.     BinarySearchTree bst;
40.     string input_s;
41.     char input;
42.     int landingTime, timeGap;
43.     string flightName;
44.
45.     cout << " Welcome to the Plane Landing System! \n";
46.     cout << DOTLINE;
47.
48.
49.     while(true){
```

Carlos Rangel
1217700029

```
50.         cout << " To begin, please initialize the time gap for landing requests: ";
51.         cin >> timeGap;
52.         if(!checkCin()){
53.             cout << " ERROR:: Invalid input!\n";
54.             cout << " Please enter an integer to initialize the time gap for landing requests.\n ";
55.         }
56.         else break;
57.     }
58.
59.     cout << " The inputted time gap is " << timeGap << "\n" << endl;
60.
61.     while(true){
62.         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
63.
64.         printMenu();
65.         cout << " Input: ";
66.         input = getchar();
67.         input = toupper(input);
68.
69.
70.         switch(input){
71.             //flush cin
72.
73.
74.             //request a landing
75.             case 'A':
76.                 cout << " Making a plane landing request. Please enter the following:\n";
77.                 cout << " Plane Flight Number: ";
78.                 cin >> flightName;
79.
80.                 //checking input
81.                 cout << " Landing Time (integers): ";
82.                 while(true){
83.                     cin >> landingTime;
84.                     if(!checkCin()){
85.                         cout << " ERROR:: Invalid input!\n";
86.                         cout << " Please enter Landing Time (Enter an integer): ";
87.                     }
88.                     else break;
89.                 }
90.
91.                 bst.treeInsert(landingTime, flightName);
92.
93.                 //checking for the timeGap (K) constraint.
94.                 if(bst.treeSuccessor(bst.treeSearch(bst.root, landingTime)) != NULL)
95.                     if( abs(bst.treeSearch(bst.root, landingTime)->key - bst.treeSuccessor(bst.treeSearch(bst.root, landingTime))-
    >key) < timeGap){
96.                         cout << "successor: " << bst.treeSearch(bst.root, landingTime)->key -
    bst.treeSuccessor(bst.treeSearch(bst.root, landingTime))->key << "\n";
97.                         cout << "\n The time gap between " << landingTime << " and flight ";
```

Carlos Rangel
1217700029

```
98.                         cout << bst.treeSuccessor(bst.treeSearch(bst.root, landingTime))->data << "(" <<
    bst.treeSuccessor(bst.treeSearch(bst.root, landingTime))->key << ") is too small!\n";
99.                         cout << " Plane request denied!\n";
100.                        bst.treeDelete(bst.treeSearch(bst.root, landingTime));
101.                        break;
102.                    }
103.
104.                if(bst.treePredecessor(bst.treeSearch(bst.root, landingTime)) != NULL)
105.                    if( abs(bst.treeSearch(bst.root, landingTime)->key - bst.treePredecessor(bst.treeSearch(bst.root,
    landingTime))->key) < timeGap){
106.                        cout << "predecessor: " << bst.treeSearch(bst.root, landingTime)->key -
    bst.treePredecessor(bst.treeSearch(bst.root, landingTime))->key << "\n";;
107.                        cout << "\n The time gap between " << landingTime << " and flight";
108.                        cout << bst.treePredecessor(bst.treeSearch(bst.root, landingTime))->data << "(" <<
    bst.treePredecessor(bst.treeSearch(bst.root, landingTime))->key << ") is too small!\n";
109.                        cout << " Plane request denied!\n";
110.                        bst.treeDelete(bst.treeSearch(bst.root, landingTime));
111.                        break;
112.                    }
113.                cout << "\n Plane " << bst.treeSearch(bst.root, landingTime)->data << " was added!\n";
114.                break;
115.
116.            //withdraw landing request
117.            //I was gonna do removal by searching landing time
118.            //but thought that would be weird in a real setting so
119.            //i overloaded treeSearch to work with inputted strings
120.            case 'B':
121.                cout << " Flight names and landing times:\n";
122.                cout << DOTLINE;
123.                bst.inOrderTreeWalk_flightName(bst.root);
124.                cout << " Please enter Landing time of the flight to remove: ";
125.                while(true){
126.                        cin >> landingTime;
127.                        if(!checkCin()){
128.                            cout << " ERROR:: Invalid input!\n";
129.                            cout << " Please enter Landing Time (Enter an integer): ";
130.                        }
131.                        else break;
132.                    }
133.
134.                if(bst.treeSearch(bst.root, landingTime) == NULL){
135.                    cout << " ERROR:: Flight Not fould!";
136.                    break;
137.                }
138.                else {
139.                    cout << " Flight " << bst.treeSearch(bst.root, landingTime)->data;
140.                    bst.treeDelete(bst.treeSearch(bst.root, landingTime));
141.                    cout << " has been deleted.\n";
142.                }
143.                break;
```

Carlos Rangel
1217700029

```
144.
145.            case 'C':
146.                cout << " Flight names and landing times:\n";
147.                cout << DOTLINE;
148.                bst.inOrderTreeWalk_flightName(bst.root);
149.                break;
150.
151.            case '?':
152.                printMenu();
153.                break;
154.
155.            case 'Q':
156.                system("pause");
157.                return 0;
158.                break;
159.
160.            default:
161.                cout << " ERROR:: Invalid input!\n";
162.                cout << " Please try again. . .\n\n";
163.                break;
164.        }
165.    }
166.}
```

167. Upload your programs for question 1 and question 2 to google drive and provide the links for your programs

Link for the program for question1

- https://drive.google.com/drive/folders/1eTQNUSk7Abf_Kzu9psHTSCVxtNmHbmQm?usp=sharing

Link for the program for question2

- https://drive.google.com/drive/folders/1z9u2QwJEHkZvqVfBzGQS2nXX_S_5TomL?usp=sharing

I've also provided the executable, as I was having compiling problems myself in case of the need to test. If you compile the .cpp files yourself, please compile using -m64 in gcc. Not sure why it doesn't do it by default.

```
PS C:\Users\chuck\OneDrive\Documents\projects\c++\hw4> g++ -m64 Landing_times_BST.cpp BinarySearchTree.cpp -o program
PS C:\Users\chuck\OneDrive\Documents\projects\c++\hw4> ./program
```

Also, for case A in Landing_time_BST.cpp, the checks for the K time gap are very big and badly formatted in this document sorry!

ALSO ALSO, I decided to get rid of the word margins, as I am assuming you have no intention of printing this page, and I believe it makes the code easier to see. Please let me know if you need a printable version of this document!