IMPLEMENTATION OF AIRPORT RUNWAY SCHEDULING SYSTEM USING AVL TREE

MINOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR

THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

Information Technology

Submitted By:

Kanishka Malhotra(1805305)

Divya Laxmi Kumari (1905419)

Sunny Kumar(1805110)

Submitted To:

Prof. Harjot Kaur Gilll

Assistant Professor

Minor Project Coordinator

Department of Information Technology

Guru Nanak Dev Engineering College,

Ludhiana-141006

Abstract

As the air transportation industry expands, airports face numerous challenges to manage the increasing traffic. Among these problems, runway crossings are a considerable source of ground traffic inefficiency and risk. Building end-around taxiways are the only strategy to avoid crossings, but these are not always feasible, and therefore airport planners must find alternatives. This study consisted of a simulation over an airport that currently requires a vast amount of its arrivals to go through runway crossings in order to reach the apron; the airport simulation software utilized was the Total Airspace and Airport Modeler (TAAM).

The airport scheduling is a challenging task where computer vision and programming language play a important role. This technology can be used in many ways such as in scheduling of any task to get fast results (example: train scheduling having single rail track or multiple, also in airplane scheduling having single runway or multiple runway)

The basic working of the project is based on principle of AVL tree and C and C++ language.

The project is all about the implementation of AVL trees in scheduling airport runway system. In which a code in C++ is written which tells us the available scheduling times for an airplane. It takes the input and return, if the tree includes the input inserted that means you are able to schedule your airplane at the given input otherwise your input is rejected by the program.

ACKNOWLEDGEMENT

We are highly grateful to Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC) Ludhiana, for providing this opportunity to carry out the minor project work on Image caption generation and translation.

The constant guidance and encouragement received from Dr. Kulwinder Singh Mann H.O.D. IT Department, GNDEC Ludhiana has been of great help in carrying out the project work and acknowledgement with reverential thanks.

We would like to place on record my deep gratitude to Prof. Parminder Kaur Wadhwa , Department of Information and Technology, GNDEC, Ludhiana for her generous guidance, help and useful suggestions.

WE express gratitude to other faculty members of the computer science and engineering department of GNDEC for their intellectual support throughout the course of this work.

Finally, We are indebted to all whosoever have contributed in this report work.

Kanishka Malhotra (1805305)

Divya Laxmi Kumari (1905419)

Sunny Kumar (1805110)

List of Figures

Contents

ABBREVIATION

TAAM- (Total Airspace and Airport Modeler)

FAA- (Federal Aviation Admistration)

1 Introduction

1.1 Introduction to Project Page

Commercial air transportation has been growing steadily in the United States and it should continue to expand. The National Forecast of Fiscal Year 2019 released by the Federal Aviation Administration (FAA) reports an expected average growth of 1.5% per year in flight operations for the next 20 years (FAA, 2019a).

To handle this growth, stakeholders in the industry are working to manage the rise in demand by increasing capacity and working on efficiency which is not enough as there is a need to handle the Airport scheduling of airplanes in order to reduce the loss of a company and passengers and for the better upliftment of the company by providing better facilities to the passengers which include saving their time by landing on time and giving them multiple flights in a day by scheduling the timings.

Since demand for air-transportation is predicted to increase, there is a need to realize additional takeoff and landing slots through better runway scheduling. In this project, we review the techniques and tools of operational research and management science that are used for scheduling aircraft landings and take-o s. The main solution techniques include AVL trees with the help of BFS and DFS algorithms.

1.2 Project Category (Internet based, Application or System Development, Research based, Industry Automation, Network or System Administration)

This project is research based as well as industry based.

- Research Based- This project is research based as we first time researched and imposed a system for runway scheduling system
- Industry based- This is also industry based as this system can be used by industries in order to maintain their runway scheduling.

1.3 Objectives

1. To study and implement AVL trees using C++.
2. To understand the performance of AVL trees by giving different number of inputs.
3. To implement a runway scheduling system for an airport with single (very busy) runway.

1.4 Identification/Reorganization of Need

This project will not only help airline industries but also help defence services as they most have single runway in their airports

- Save Time: Scheduling od airplanes will save a lot of time of the passengers as well as the user.
- Save Land: This will also save your land as you can use only a single runway efficiently.
- Reduce Mishappenings: This will reduce mishappening happen during uncertain landings and takeoffs like a accident happened in

1.5 Existing System

Railway scheduling system is existing already which is proposed for the scheduling of trains in order to avoid the waiting time period for other trains for the platforms tracks on the railway stations.

1.6 Proposed System

Runway scheduling System is proposed for controlling the air traffic and for handling the timings of airplanes

1.7 Unique Features of the System

- Implemented AVL trees for the first time in Runway scheduling

- Proposed the system for single runway.

- Timestamp in the system is useful in maintaining the traffic .

2 Requirement Analysis and System Specification

2.1 Feasibility Study (Technical, Economical, Operational)

The project is technically feasible as we implemented the AVL trees using C++ language. also this is economically feasible because the software used in this project are open source. This project is socially feasible too as the implementation of airport runway scheduling can be used by the airlines industries for the better services for their passengers

2.2 Software Requirement Specification Document which must include the following: (Data Requirement, Functional Requirement, Performance Requirement, Dependability Requirement, Maintainability requirement, Security Requirement, Look and feel requirement)

2.2.1 Introduction

Our runway scheduling project is based on AVLtree which is utilized in data structure. In this project, we review the techniques and tools of operational research and management science that are used for scheduling flights landings and take-offs. The main solution techniques include AVL trees with the help of BFS and DFS algorithms.

Figure 1: Airport with single Runway

2.2.2 Organisation Needs

- Advance runway system.
- Efficient and compact runway.
- Single runway is more efficient due to timestamp.

2.2.3 Intended Audience

It provides a clear picture of potential growth drivers, constraints, competitive landscape, segment analysis, and market size by country and area.

2.2.4 Intended Use

It is used for both military aviation and civil aviation. They typically contain facilities of both a civil airport and a military base.

2.2.5 Scope of development

The scope of this project is to develop the single runway scheduling system. To avoid the risk of flight clash and optimize ongoing challenge for air traffic controllers

2.2.6 Purpose

The ever-growing increase in air traffic is challenging airports regarding dealing with complex challenges of airside infrastructure. Runway crossings generate delay for arrivals and delay for departures. Since, the baseline validation included validating the flight schedule, once the baseline was validated, the alternative flight schedule will also created. TAAM which automatically generate schedules based on an original schedule.

2.2.7 System Features and Requirements –

Functional Requirements

- Manage and avoid the risk of flight clashes.
- Use of timestamp.
- To control the Air Traffic Congestion.

2.3 Expected hurdles

- Inappropriate arrival of flight
- Inappropriate departure of flight

- Collison of two flight.

2.4 SDLC Model to be used

SDLC model used is Spiral model: -

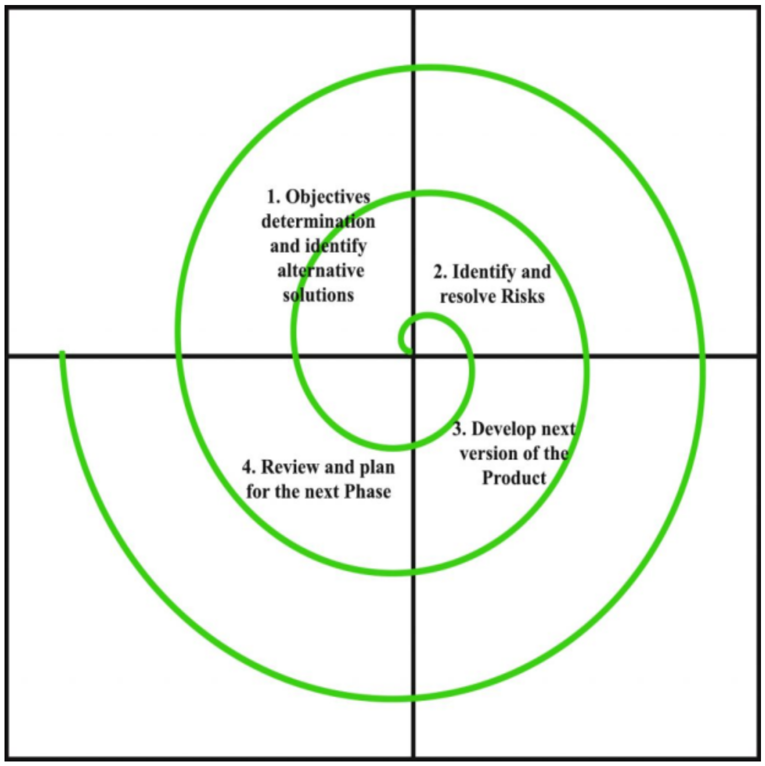

1. Objectives determination and identify alternative solutions
2. Identify and resolve Risks
3. Develop next version of the Product
4. Review and plan for the next Phase

Figure 2: Spiral Model

Spiral Model is a risk-driven software development process model. It is a combination of waterfall model and iterative model. Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.

3 System Design

3.1 Design Approach (Function oriented or Object oriented)

Class Diagram is shown below for detailed design concept :-

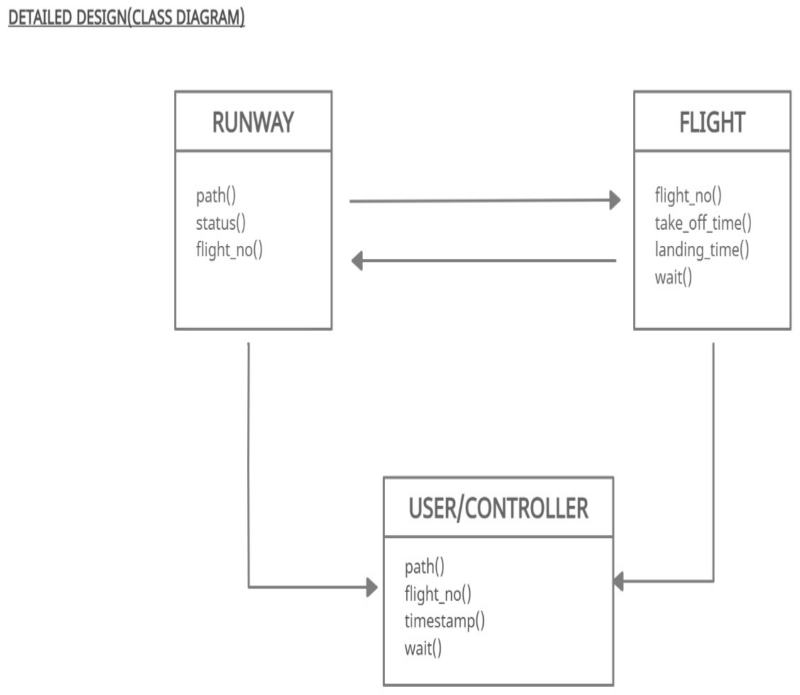


Figure 3: Class Diagram

3.2 System Design using various structured analysis and design tools such as: DFD's, Data Dictionary, Structured charts, Flowcharts or UML

3.2.1 Data Flow Diagram :-

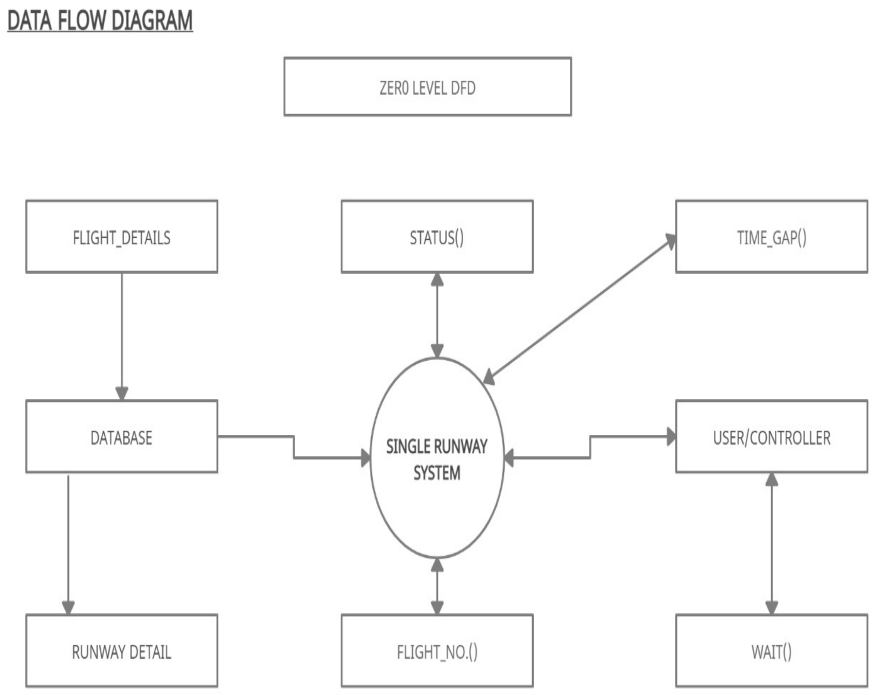


Figure 4: Data Flow Diagram

3.2.2 UML diagram :-

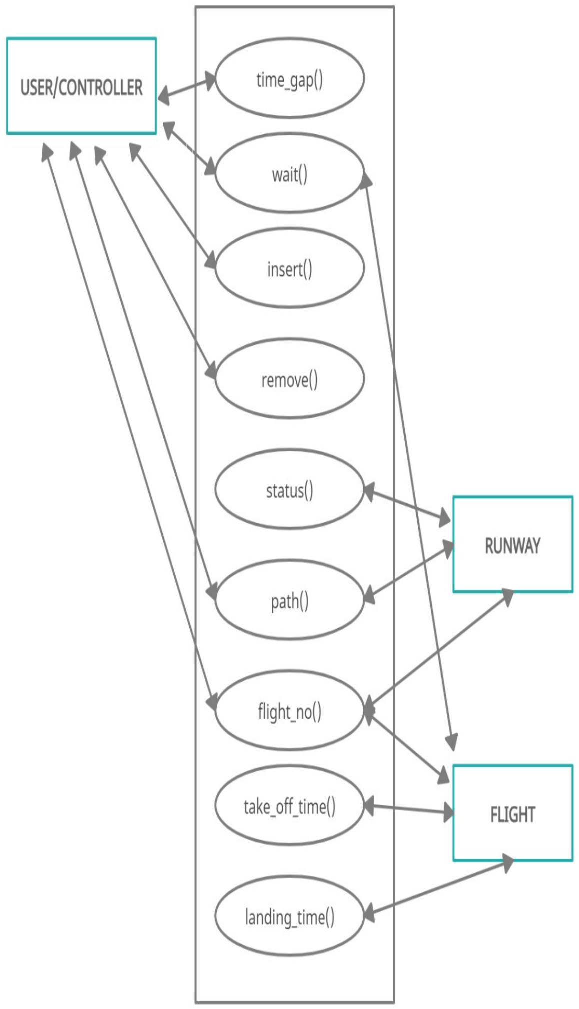


Figure 5: UML Diagram

3.2.3 Flow chart

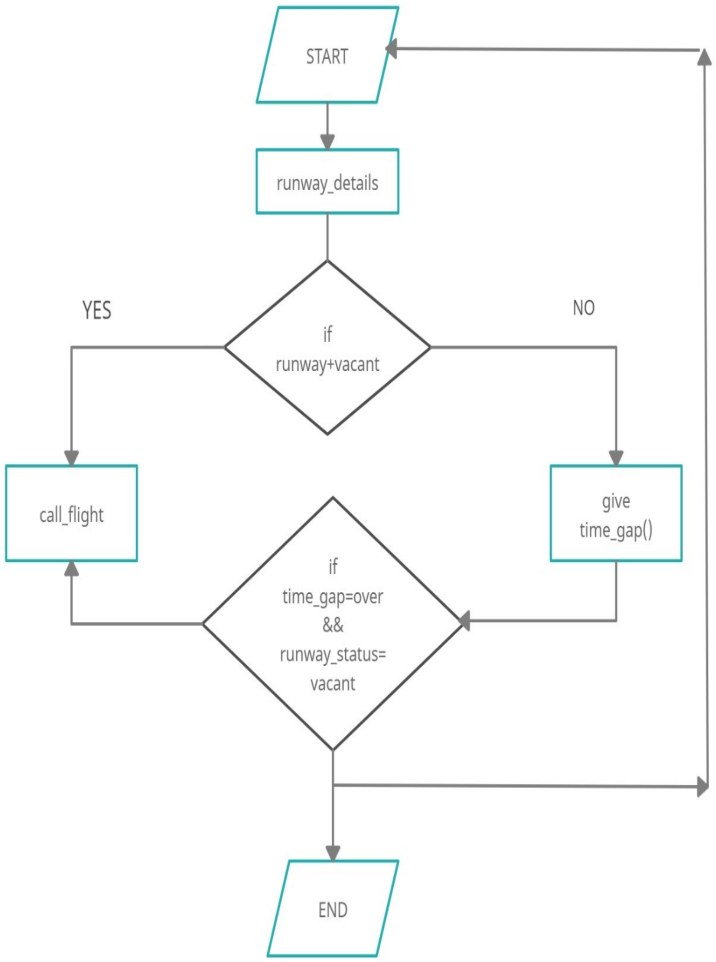


Figure 6: Flow chat

3.3 Database Design

3.3.1 ER Diagrams

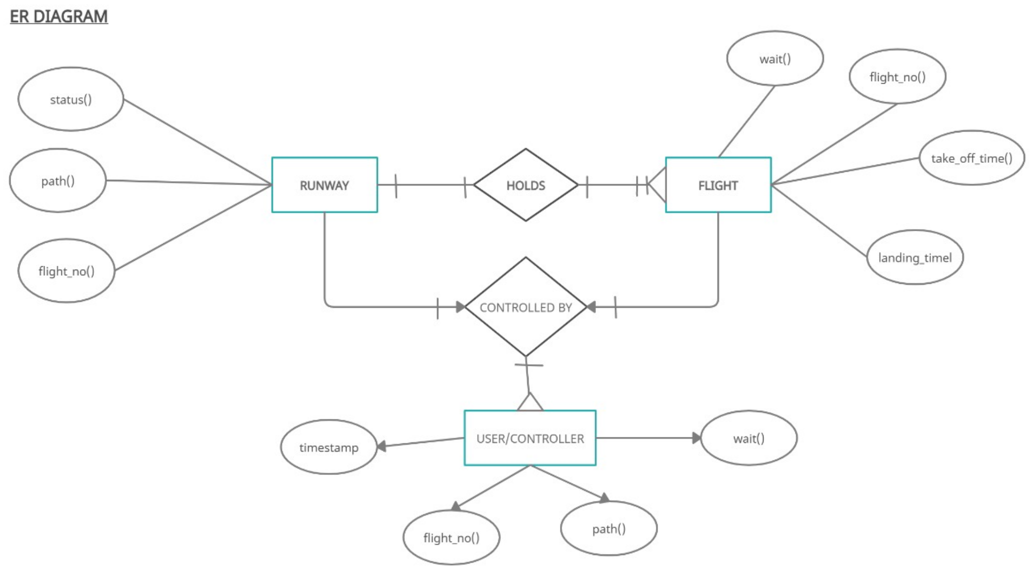


Figure 7: ER Diagram

3.4 Methodology

Firstly we studied AVL tree using C ++ language and after this we did case study on Airports -Runway scheduling system and implemented this system using AVL tree and we saw the outputs of this system by giving different inputs to the system in order to avoid risk.

4 Implementation, Testing, and Maintenance

4.1 Introduction to Languages, IDE's, Tools and Technologies used for Implementation

In our project we use C++ Language and used code blocks for implementation.

4.2 Coding standards of Language used

- C++ is a highly portable language and is often the language of selection for multi-device, multi-platform app development.
- C++ is an object-oriented programming language and includes concepts like classes, inheritance, polymorphism, data abstraction and encapsulation which allow code reusability and makes programs very maintainable.
- C++ use multi-paradigm programming. The Paradigm means the style of programming. Paradigm concerned about logics, structure, and procedure of the program. C++ is multi-paradigm means it follows three paradigm Generic, Imperative, Object Oriented.
- It is useful for the low-level programming language and very efficient for general purpose.
- C++ gives the user complete control over memory management. This can be seen both as an advantage and a disadvantage as this increases the responsibility of the user to manage memory rather than it being managed by the Garbage collector.
- The wide range of applications – From GUI applications to 3D graphics for games to real-time mathematical simulations, C++ is everywhere.
- C++ has a huge community around it. Community size is important, because the larger a programming language community, more is the more support you would be likely to get. C++ is the 6th most used and followed tag on Stack Overflow and GitHub.
- C++ has a very big job market as it is used in various industries like finance, app development, game development, Virtual reality, etc.
- C++'s greatest strength is how scalable it could be, so apps that are very resource intensive are usually built with it. As a statically written language, C++ is usually more performant than the dynamically written languages because the code is type-checked before it is executed.

4.3 Compatibility with C –

C++ is compatible with C and virtually every valid C program is a valid C++ program.

4.4 Testing Techniques and Test Plans

4.4.1 Uuit Testing

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. TestLeft is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

4.4.2 Testing Plans .

1. Testing is done by using diffrent input for a program.

2. Difference platform are used for testing weather our program is running on every platform or not.

5 Results and Discussions

5.1 Brief Description of various modules in the system

Our System is consist of a code written in C++ language the functions used in that code are:

Insert Function

In this function we insert a value x and imagine a pointer t and checks whether the pointer t is null or not if null than we insert a value at t and consider it as a root node and if t is not null than we check that whether the value x is less than x-10 or x+10 and respectively we insert the data as a left or right sub node respectively or else we reject the value.

SingleRightRotation

Function In this function we have data at left sub node, right sub node and at root and we create a pointer u, firstly we insert the value of left sub node at u pointer, secondly we insert the value of right sub node at left sub node and lastly we insert the value of root node at right sub node.

SingleLeftRotation Function

In this function we have data at left sub node, right sub node and at root and we create a pointer u, firstly we insert the value of right sub node at u pointer,secondly we insert the value of left sub node at right sub node and lastly we insert the value of root node at left sub node.

DoubleLeftRotation Function

As the function name defines double rotation than what we used to do is we apply a SingleRightRotation at right sub node and after the rotation apply SingleLeftRotation on the right sub node and return the value of it.

DoubleRightRotation Function

As the function name defines double rotation than what we used to do is we apply a SingleLeftRotation at left sub node and after the rotation apply SingleRightRotation on the left sub node and return the value of it.

FindMin Function

FindMin function is used to find the minimum value of the tree for which we check the value at t if it is null than we return the value as null or else we check the left sub node of t if it is null than we return the t as minimum value or we consider the left value as t and again start the process.

FindMax Function

FindMax function is used to find the maximum value of the tree for which we check the value at t if it is null than we return the value as null or else we check the right sub node of t if it is null than we return the t as minimum value or we consider the right value as t and again start the process.

Remove Function

Remove function is used to check whether the value inserted is suitable for the output as AVL tree with a timestamp of 10 min or not foir which we first check the value of t if it is null we return as null or else we check the data is less than t value if yes we apply the remove function of the left nodes or vice-versa.

Height Function

The Height Function is used to find the height of the AVL tree if t is null it return the value as -1 else it return the height of tree.

GetBalance Function

This function is used to find the Balance Factor of a AVL tree in which we check whether the t is null or not if yes return the value 0 or we find the difference of left sub node and right sub node

Inorder Function

In this function we used inorder traversal for printing the AVL tree in which we check whether t is null or not if yes than we return as null or else we print the left sub nodes and than the root node and than right sub nodes which is a format of inorder traversal

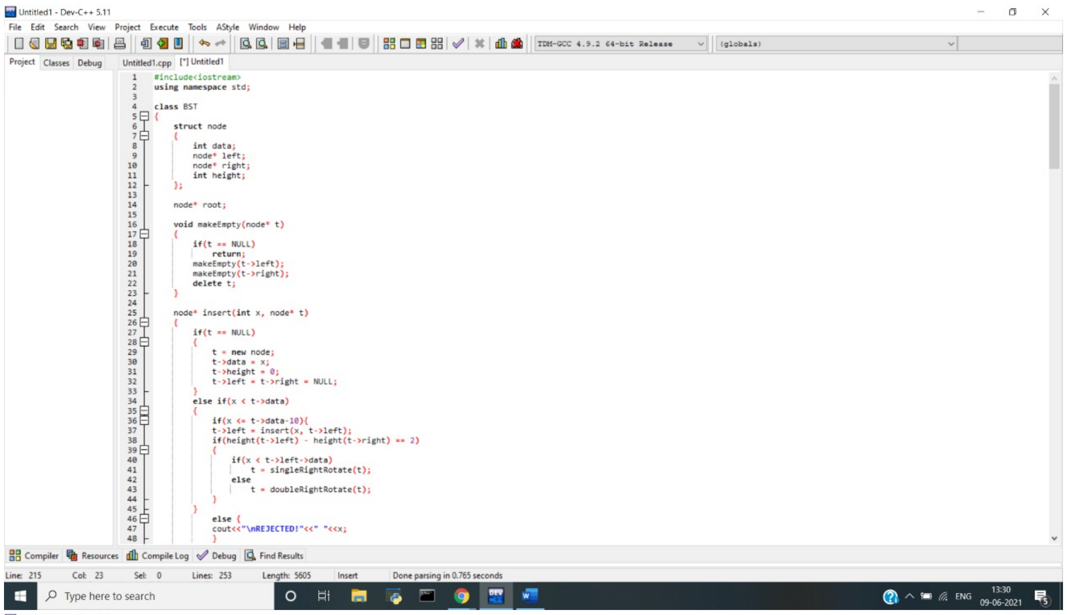
Display function

This function is used to display the AVL tree in inordered form.

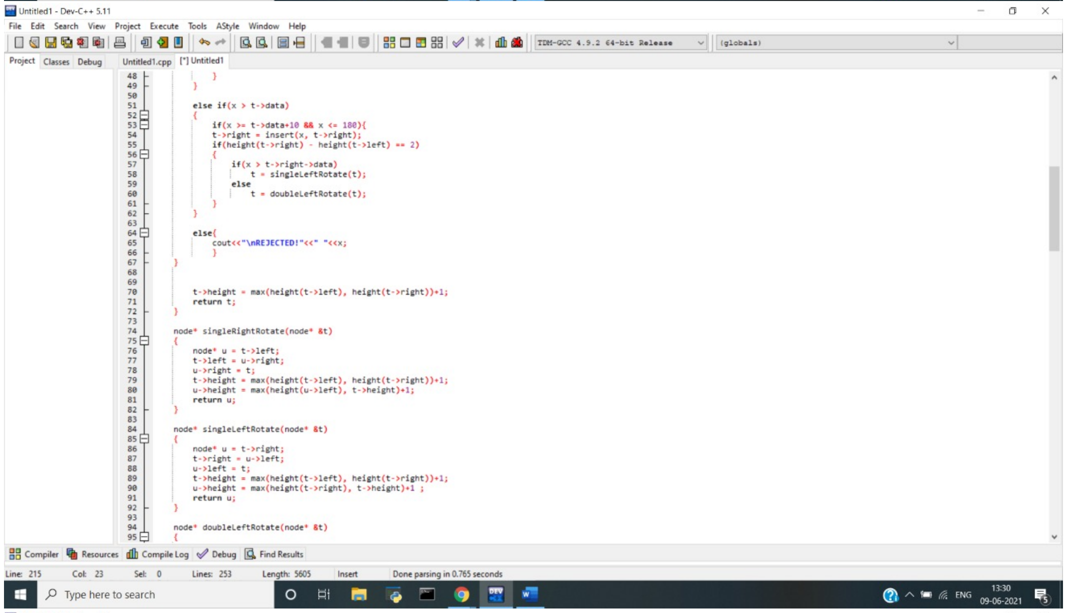
5.2 Snapshots of system with brief detail of each

5.2.1 Snapshot of the code:

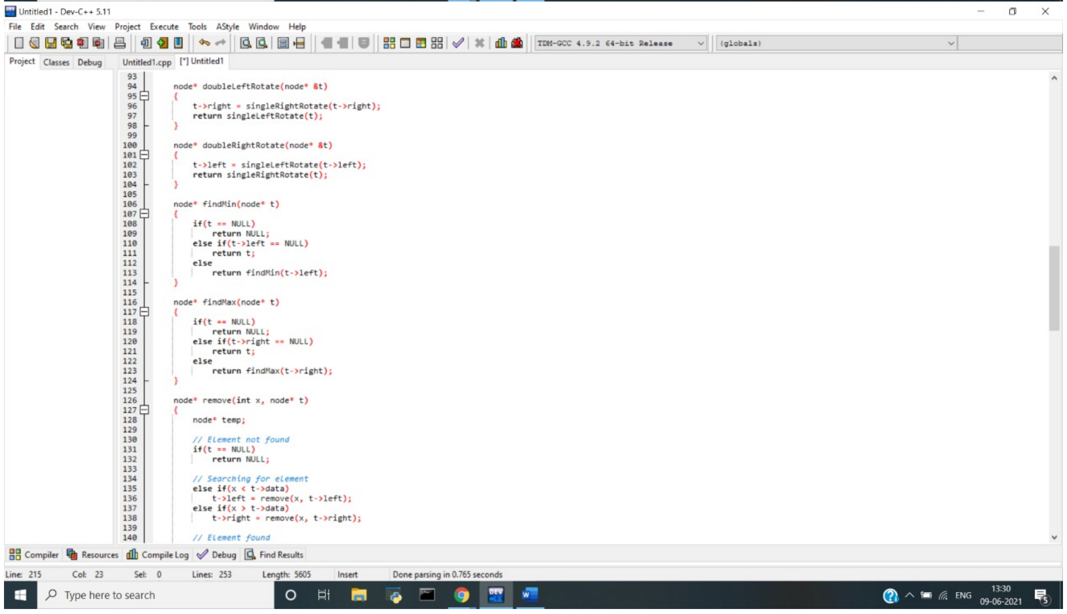
....



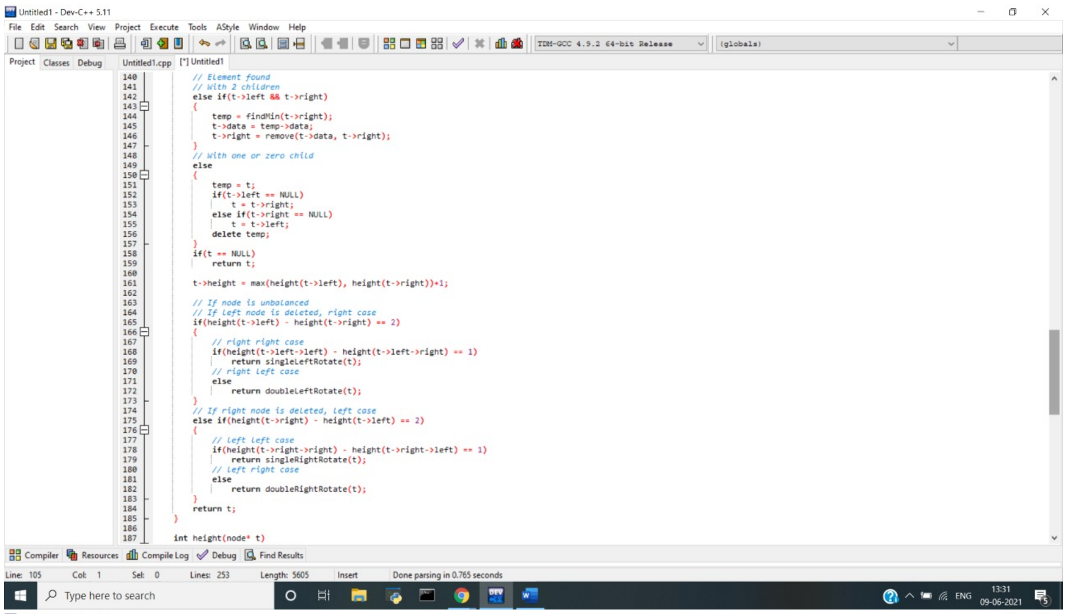
```
#include<iostream>
using namespace std;

class BST
{
    struct node
    {
        int data;
        node* left;
        node* right;
        int height;
    };

    node* root;

    void makeEmpty(node* t)
    {
        if(t == NULL)
            return;
        makeEmpty(t->left);
        makeEmpty(t->right);
        delete t;
    }

    node* insert(int x, node* t)
    {
        if(t == NULL)
        {
            t = new node;
            t->data = x;
            t->height = 0;
            t->left = t->right = NULL;
        }
        else if(x < t->data)
        {
            if(x <= t->data-10){
            t->left = insert(x, t->left);
            if(height(t->left) - height(t->right) == 2)
            {
                if(x < t->left->data)
                    t = singleRightRotate(t);
                else
                    t = doubleRightRotate(t);
            }
        }
            else {
            cout<<"\nREJECTED!"<<" "<<x;
            }
```



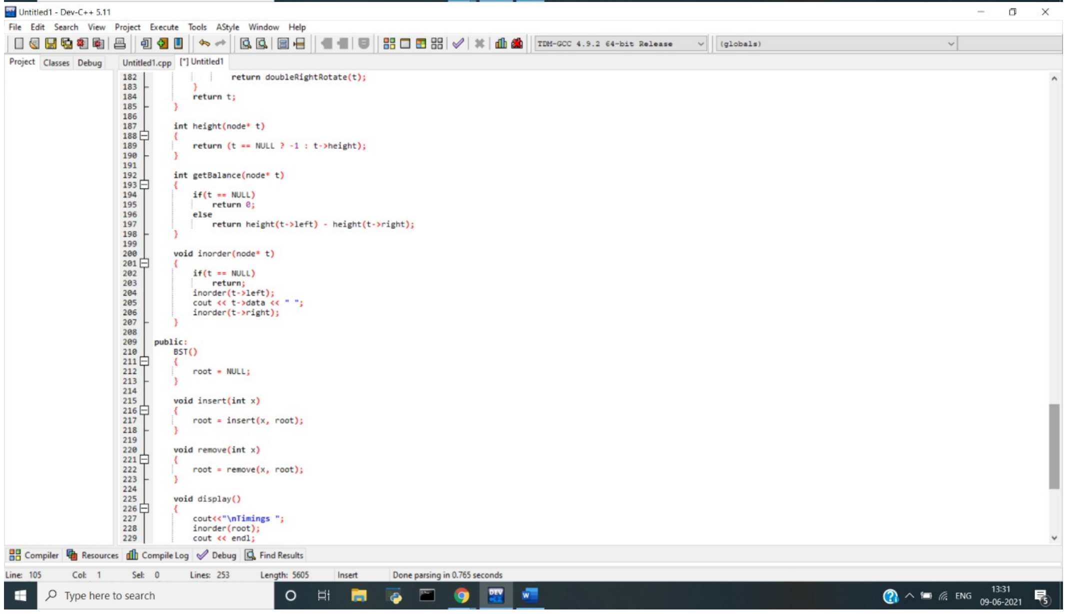
```
            }
        }

        else if(x > t->data)
        {
            if(x >= t->data+10 && x <= 100){
            t->right = insert(x, t->right);
            if(height(t->right) - height(t->left) == 2)
            {
                if(x > t->right->data)
                    t = singleLeftRotate(t);
                else
                    t = doubleLeftRotate(t);
            }
        }

        else{
            cout<<"\nREJECTED!"<<" "<<x;
            }
    }


        t->height = max(height(t->left), height(t->right))+1;
        return t;
    }

    node* singleRightRotate(node* &t)
    {
        node* u = t->left;
        t->left = u->right;
        u->right = t;
        t->height = max(height(t->left), height(t->right))+1;
        u->height = max(height(u->left), t->height)+1;
        return u;
    }

    node* singleLeftRotate(node* &t)
    {
        node* u = t->right;
        t->right = u->left;
        u->left = t;
        t->height = max(height(t->left), height(t->right))+1;
        u->height = max(height(t->right), t->height)+1 ;
        return u;
    }

    node* doubleLeftRotate(node* &t)
    {
```



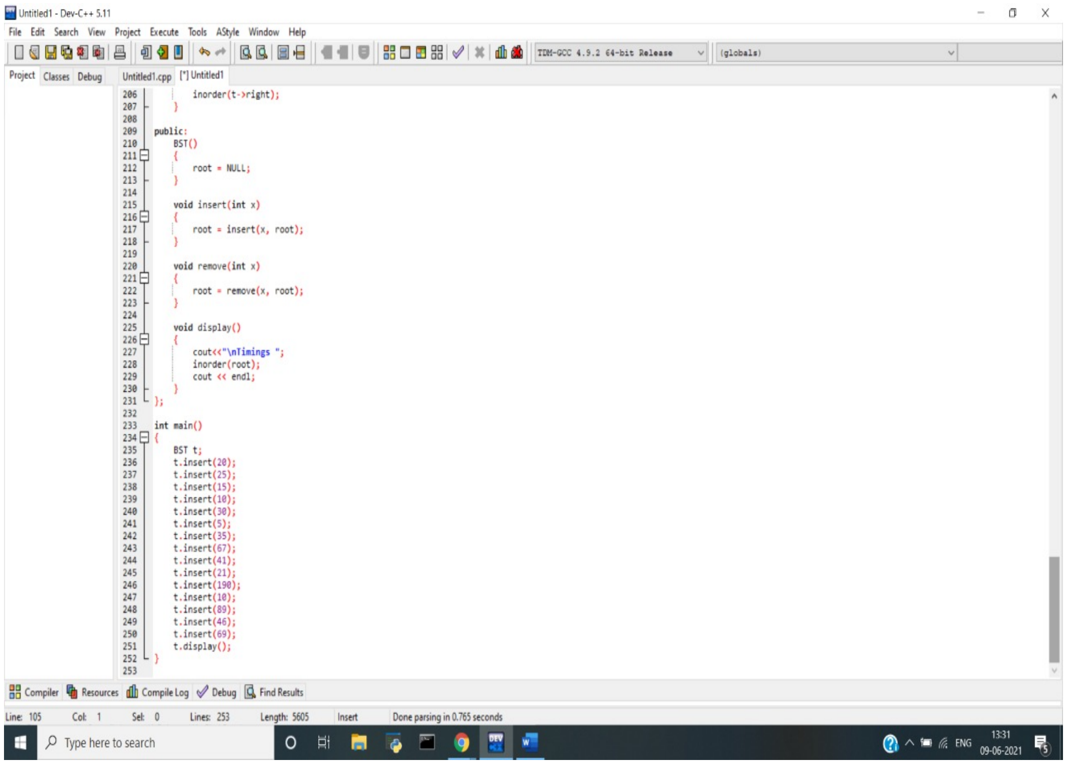
```

    node* doubleLeftRotate(node* &t)
    {
        t->right = singleRightRotate(t->right);
        return singleLeftRotate(t);
    }

    node* doubleRightRotate(node* &t)
    {
        t->left = singleLeftRotate(t->left);
        return singleRightRotate(t);
    }

    node* findMin(node* t)
    {
        if(t == NULL)
            return NULL;
        else if(t->left == NULL)
            return t;
        else
            return findMin(t->left);
    }

    node* findMax(node* t)
    {
        if(t == NULL)
            return NULL;
        else if(t->right == NULL)
            return t;
        else
            return findMax(t->right);
    }

    node* remove(int x, node* t)
    {
        node* temp;

        // Element not found
        if(t == NULL)
            return NULL;

        // Searching for element
        else if(x < t->data)
            t->left = remove(x, t->left);
        else if(x > t->data)
            t->right = remove(x, t->right);

        // Element found
```



```
            // Element found
            // With 2 children
            else if(t->left && t->right)
            {
                temp = findMin(t->right);
                t->data = temp->data;
                t->right = remove(t->data, t->right);
            }
            // With one or zero child
            else
            {
                temp = t;
                if(t->left == NULL)
                    t = t->right;
                else if(t->right == NULL)
                    t = t->left;
                delete temp;
            }
            if(t == NULL)
                return t;

            t->height = max(height(t->left), height(t->right))+1;

            // If node is unbalanced
            // If left node is deleted, right case
            if(height(t->left) - height(t->right) == 2)
            {
                // right right case
                if(height(t->left->left) - height(t->left->right) == 1)
                    return singleLeftRotate(t);
                // right left case
                else
                    return doubleLeftRotate(t);
            }
            // If right node is deleted, left case
            else if(height(t->right) - height(t->left) == 2)
            {
                // left left case
                if(height(t->right->right) - height(t->right->left) == 1)
                    return singleRightRotate(t);
                // left right case
                else
                    return doubleRightRotate(t);
            }
            return t;
        }

        int height(node* t)
```



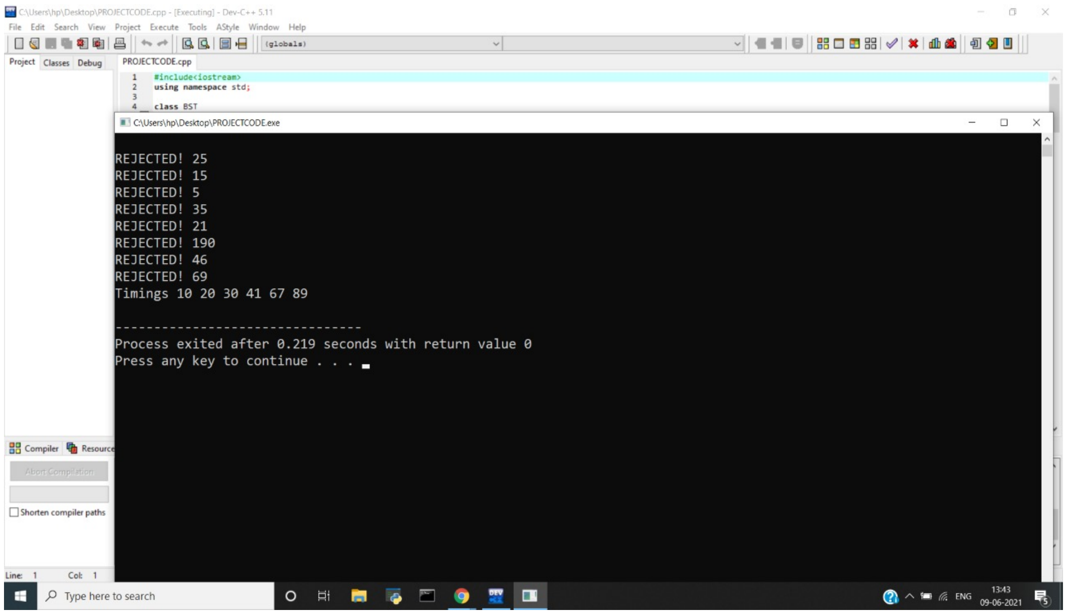
```
                    return doubleRightRotate(t);
            }
            return t;
        }

        int height(node* t)
        {
            return (t == NULL ? -1 : t->height);
        }

        int getBalance(node* t)
        {
            if(t == NULL)
                return 0;
            else
                return height(t->left) - height(t->right);
        }

        void inorder(node* t)
        {
            if(t == NULL)
                return;
            inorder(t->left);
            cout << t->data << " ";
            inorder(t->right);
        }

    public:
        BST()
        {
            root = NULL;
        }

        void insert(int x)
        {
            root = insert(x, root);
        }

        void remove(int x)
        {
            root = remove(x, root);
        }

        void display()
        {
            cout<<"\nTimings ";
            inorder(root);
            cout << endl;
```



```
            inorder(t->right);
        }

public:
    BST()
    {
        root = NULL;
    }

    void insert(int x)
    {
        root = insert(x, root);
    }

    void remove(int x)
    {
        root = remove(x, root);
    }

    void display()
    {
        cout<<"\nTimings ";
        inorder(root);
        cout << endl;
    }
};

int main()
{
    BST t;
    t.insert(20);
    t.insert(25);
    t.insert(15);
    t.insert(10);
    t.insert(30);
    t.insert(5);
    t.insert(35);
    t.insert(67);
    t.insert(41);
    t.insert(21);
    t.insert(190);
    t.insert(10);
    t.insert(89);
    t.insert(46);
    t.insert(69);
    t.display();
}
```

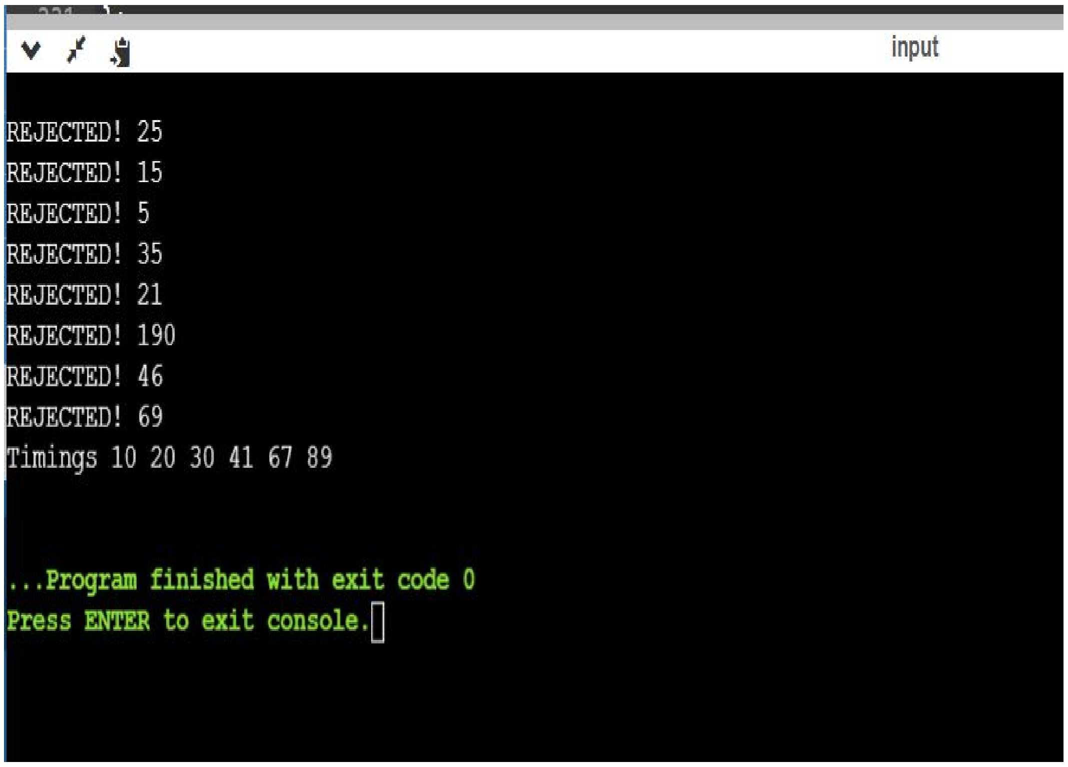
Figure 8: Snapshots of Output

5.2.2 Snapshot of the output using diff inputs:

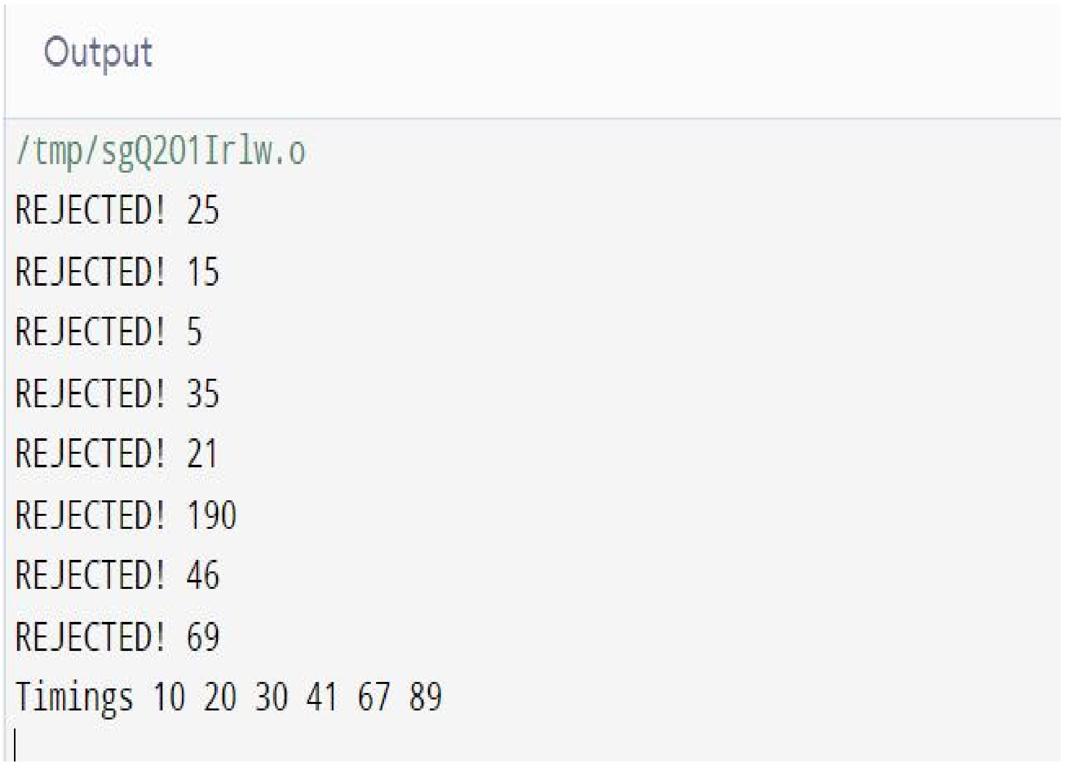


```
#include<iostream>
using namespace std;

class BST
```

```
REJECTED! 25
REJECTED! 15
REJECTED! 5
REJECTED! 35
REJECTED! 21
REJECTED! 190
REJECTED! 46
REJECTED! 69
Timings 10 20 30 41 67 89

--------------------------------
Process exited after 0.219 seconds with return value 0
Press any key to continue . . .
```



```
REJECTED! 25
REJECTED! 15
REJECTED! 5
REJECTED! 35
REJECTED! 21
REJECTED! 190
REJECTED! 46
REJECTED! 69
Timings 10 20 30 41 67 89


...Program finished with exit code 0
Press ENTER to exit console.
```



```
/tmp/sgQ201Irlw.o
REJECTED! 25
REJECTED! 15
REJECTED! 5
REJECTED! 35
REJECTED! 21
REJECTED! 190
REJECTED! 46
REJECTED! 69
Timings 10 20 30 41 67 89
```

Figure 9: Snapshots of Output

6 Conclusion and Future Scope

6.1 Conclusion

This project is totally based on Object Oriented language i.e C++ and AVL tree are used to implement the Runway Scheduling System.

In this project First we take the input by the programmer and give them a output with the inserted item and the AVL tree output with a timestamp used in the program for the safe landing and take-off of airplanes.Timestamp is of 10 min which is a maximum time for a plane to land and take-off.Also,this system is for single runway and the timmings provided by the program tells the pilot that the proposed time inserted by you is available for landing or not.If not you can enter another inputs in order to check the availability .

6.2 Future scope

In the upcoming era Commercial air transportation will increase steadily and it should continue to expand. The National Forecast of Fiscal Year 2019 released by the Federal Aviation Administration (FAA) reports an expected average growth of 1.5% per year in flight operations for the next 20 years (FAA, 2019a).For handling all this we will need a system to define the scheduling time and the airline industries will need this for their better upliftment and for their security issues.

So,there is so much scope of this project in the airline industries with single and busy runway

References

[1] Tom.Reding, "Aircraft Takeoff and Landing Area": Available at:
https://en.m.wikipedia.org/wiki/Runway

[2]Mandarax, B(february 2017). "Spiral Model": Available at:
https://en.m.wikipedia.org/wiki/Spiral_model

[3]Noel Wurst, Sr. Manager Communications, SmartBear,"Software Testing Methodologies",Available at:
https://smartbear.com/learn/automated-testing/software-testing-methodologies/

[4]Kumar Varma, B(14 Febraury 2018),"Avantages of C++" ; Available at:
https://www.tutorialspoint.com/What-are-the-advantages-of-Cplusplus-Programming-Language