SECJ2013-04 DATA STRUCTURE & ALGORITHM

SEMESTER 1, 2025/2026

LECTURER:

MS.LIZAWATI BINTI MI YUSUF

MINI PROJECT:

SCHOOL CO-CURRICULAR ACTIVITIES

SUBMISSION DATE : 11 JANUARY 2026

GROUP 2

NAME	MATRIC NUMBER
ADLYN NATASYA BINTI AZNUL RIZAL	A24CS0032
HANIS SOFIYA BINTI BADRUL EHSAN	A24CS0080
SITI MAISARAH BINTI AHMAD ARZMY	A24CS0190

Table of Content

1.0 Synopsis

The School Co-Curricular Activities Management System is developed to assist schools in recording, organising and managing student participation in co-curricular activities efficiently. The system applies appropriate data structure concepts which are a queue and binary search tree for systematic data handling and effective record management.

The system stores student information such as name, class, IC number, activity type, activity name, position held and attendance. Users are able to add new student records, display forward or backward the list of students, search for student information based on IC number and delete student records. The queue data structure is used to manage deletion operations in a First-In-First-Out (FIFO) manner, making sure that the records are removed sequentially.

Other than that, a binary search tree is implemented to store student records using the IC number as the key. This allows efficient searching, insertion and deletion of student records. The system ensures data consistency by removing corresponding records from both queue and binary search tree during deletion.

In conclusion, this system provides structured and user friendly solutions for managing co-curricular records by implementing queue and binary search tree data structures. Thus, the system ensures organized records, quick data retrieval and orderly deletion to support school administration.

2.0 Objective

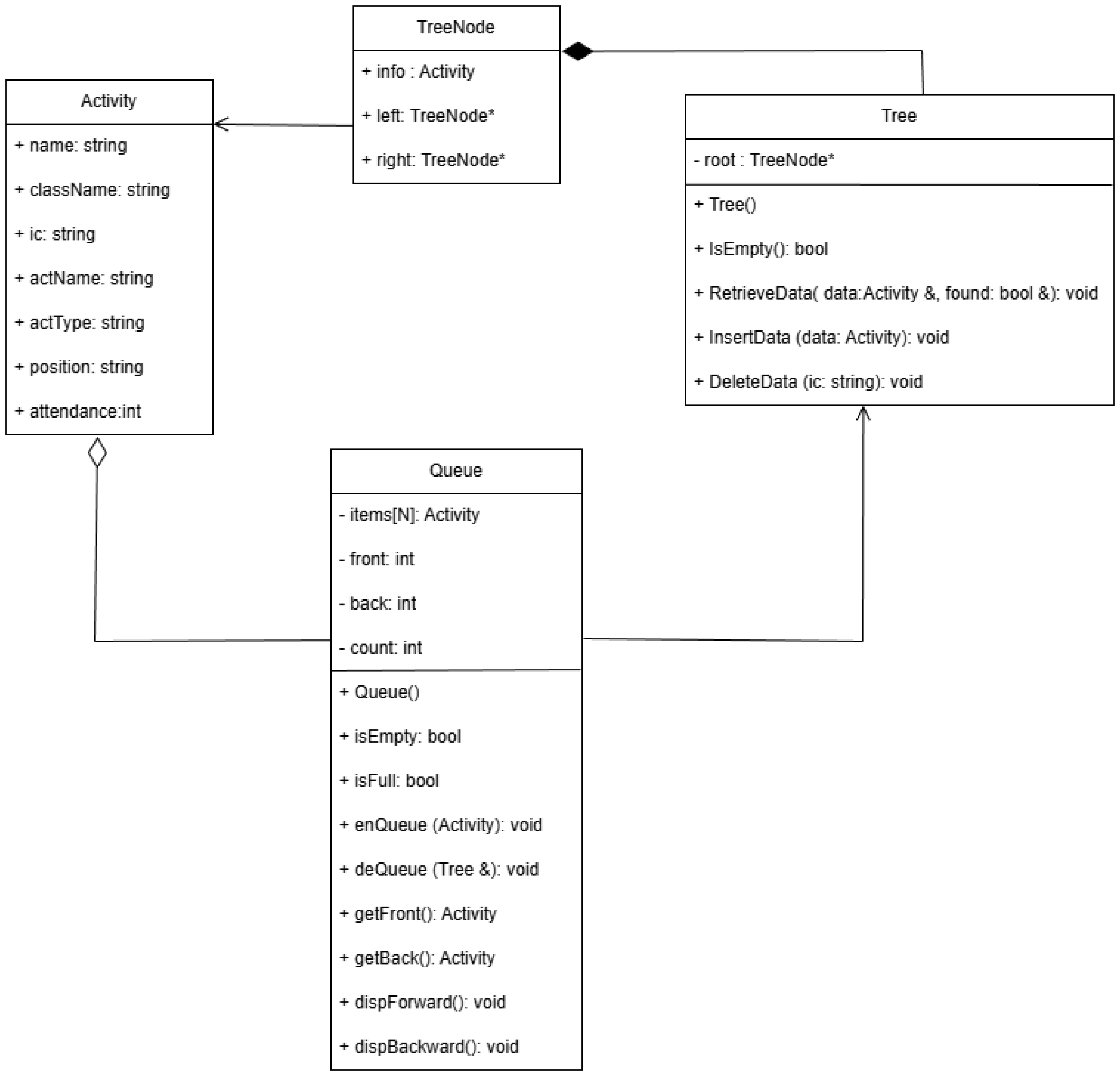
The main objective of this system is to help schools manage and organise student co-curricular activity records in a more efficient way. This system is designed to store important student details such as name, class, IC number, activity type, activity name, position held, and total attendance. By using a queue data structure, the system ensures that student records are removed in a First-In-First-Out (FIFO) order, while the binary search tree allows users to quickly search for student records based on their IC number. Overall, this system aims to reduce manual work, improve data accuracy, and make it easier for school staff to handle co-curricular information.

3.0 Scope

The scope of this system focuses on managing student co-curricular activity records in a school environment. The system allows users to add new student details, view the list of students in either forward or backward order, search for students using their IC numbers, and delete student records in a First-In-First-Out sequence using a queue. A binary search tree is implemented to make searching, inserting, and deleting records faster and more efficient. This system only covers co-curricular activities such as clubs, sports, and uniformed bodies, along with student positions and attendance. It does not include academic subjects, exam results, or other school administration features outside co-curricular management.

4.0 Class Design

4.1 Class Diagram

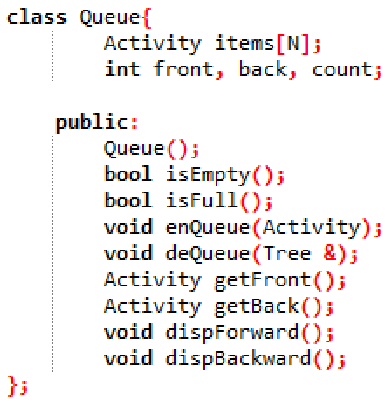


5.0 Data Structure Implementation

In this School Co-Curricular Activities Management System, two main data structure concepts are used, which are the Queue and the Binary Search Tree (BST). Each of these data structures plays an important role in managing student records efficiently.

5.1 Queue

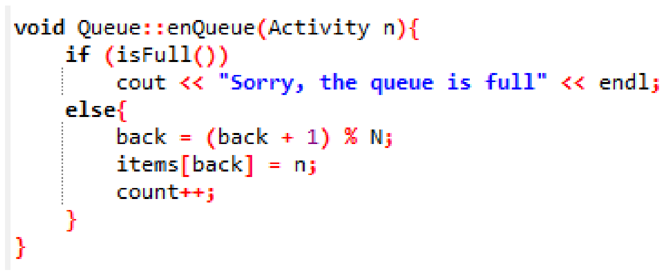
The Queue is used to handle student records in a First-In-First-Out (FIFO) manner.



```
class Queue{
        Activity items[N];
        int front, back, count;

    public:
        Queue();
        bool isEmpty();
        bool isFull();
        void enQueue(Activity);
        void deQueue(Tree &);
        Activity getFront();
        Activity getBack();
        void dispForward();
        void dispBackward();
};
```

a) enQueue function

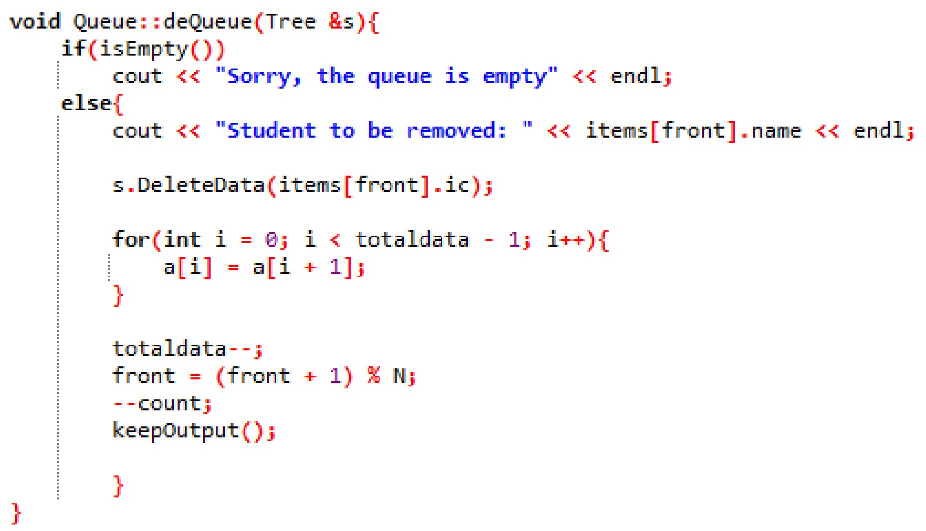
When student data is read from the file or added through the system, it is inserted into the queue using the enQueue() function. This means the first student added will be placed at the front of the queue.



```
void Queue::enQueue(Activity n){
    if (isFull())
        cout << "Sorry, the queue is full" << endl;
    else{
        back = (back + 1) % N;
        items[back] = n;
        count++;
    }
}
```

b) deQueue function

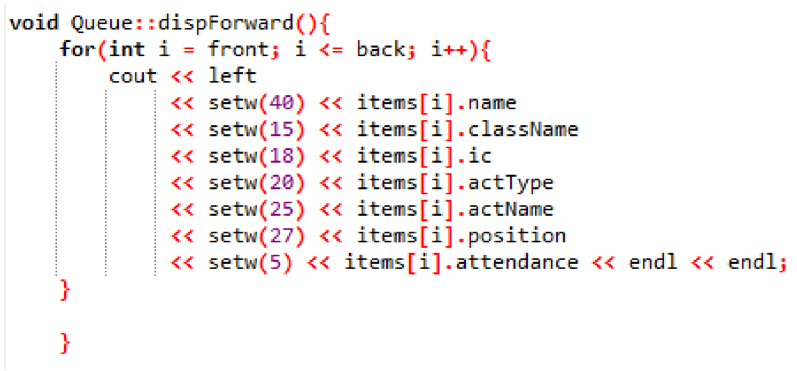
When a student record needs to be removed, the deQueue() function is used, which removes the student at the front of the queue. This ensures that records are deleted in the same order they were added, which keeps the data organised and fair. At the same time, when a record is deleted from the queue, the same student is also removed from the binary search tree and the file, so all data structures stay consistent. Increment of front using modular arithmetic and decrement of count shows circular array implementation.



```
void Queue::deQueue(Tree &s){
    if(isEmpty())
        cout << "Sorry, the queue is empty" << endl;
    else{
        cout << "Student to be removed: " << items[front].name << endl;

        s.DeleteData(items[front].ic);

        for(int i = 0; i < totaldata - 1; i++){
            a[i] = a[i + 1];
        }

        totaldata--;
        front = (front + 1) % N;
        --count;
        keepOutput();

        }
}
```

c) dispForward function

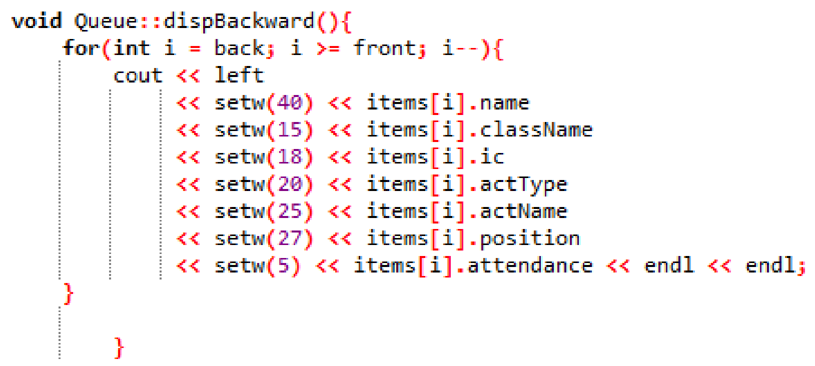
The function uses for loop to display student information from front to back to show the order of the student list stored in Queue.



```
void Queue::dispForward(){
    for(int i = front; i <= back; i++){
        cout << left
             << setw(40) << items[i].name
             << setw(15) << items[i].className
             << setw(18) << items[i].ic
             << setw(20) << items[i].actType
             << setw(25) << items[i].actName
             << setw(27) << items[i].position
             << setw(5) << items[i].attendance << endl << endl;
    }

    }
```

d) dispBackward function

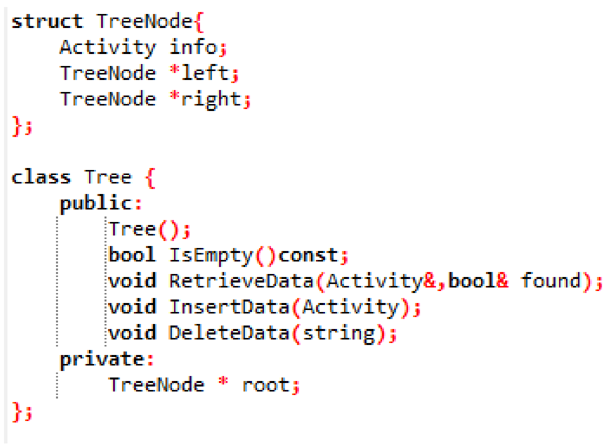
The function uses for loop to display student information from back to front to show the order of the student list stored in Queue.



```
void Queue::dispBackward(){
    for(int i = back; i >= front; i--){
        cout << left
             << setw(40) << items[i].name
             << setw(15) << items[i].className
             << setw(18) << items[i].ic
             << setw(20) << items[i].actType
             << setw(25) << items[i].actName
             << setw(27) << items[i].position
             << setw(5) << items[i].attendance << endl << endl;
    }

        }
```

5.2 Tree

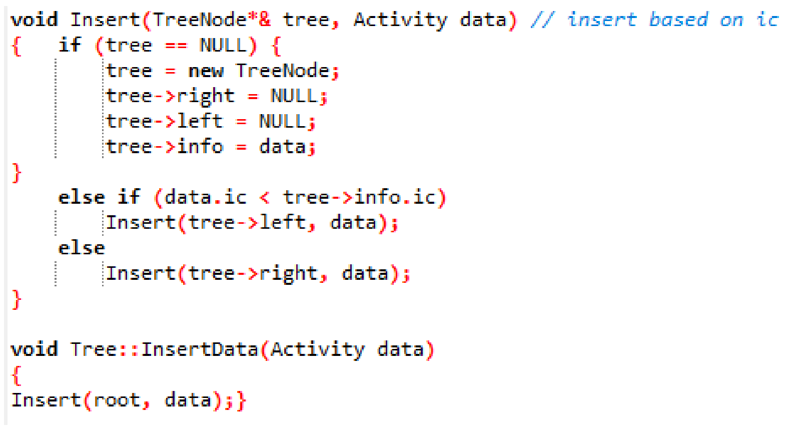
The Binary Search Tree (BST) is used to search student records based on their IC numbers.



```
struct TreeNode{
    Activity info;
    TreeNode *left;
    TreeNode *right;
};

class Tree {
    public:
        Tree();
        bool IsEmpty()const;
        void RetrieveData(Activity&,bool& found);
        void InsertData(Activity);
        void DeleteData(string);
    private:
        TreeNode * root;
};
```

a) InsertData function

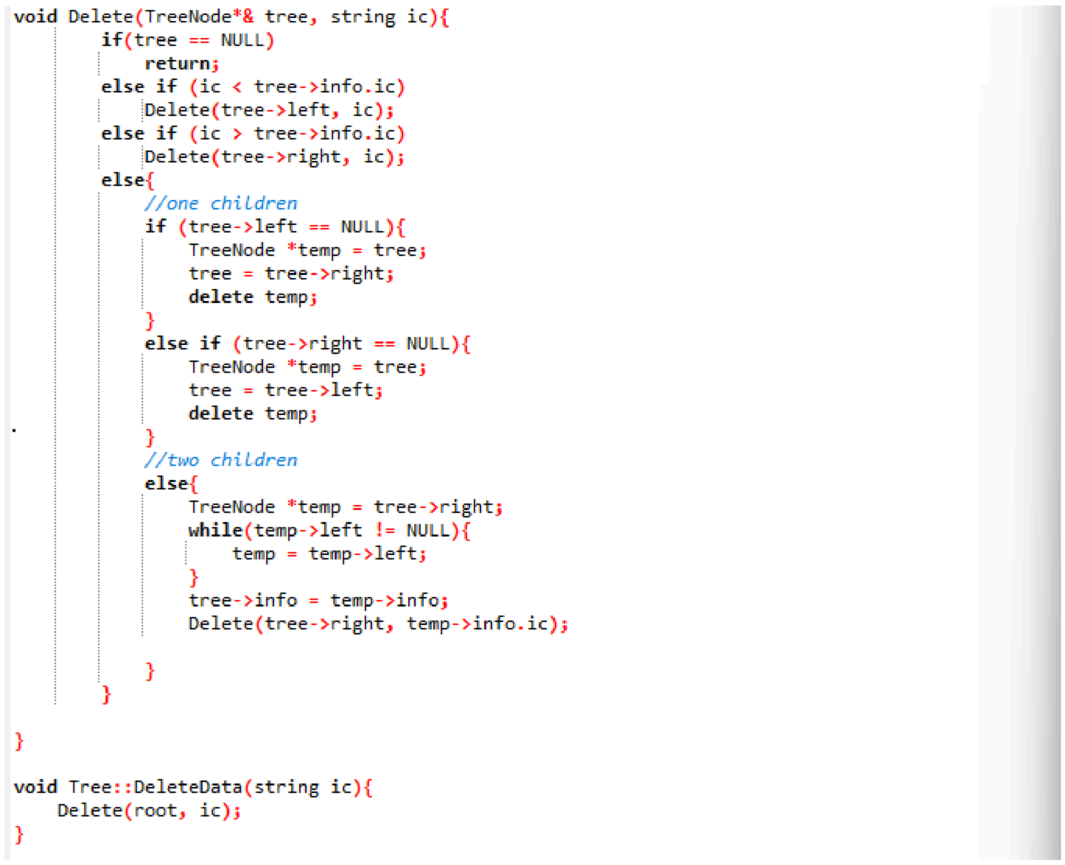
Each student's IC number is treated as a key, and the records are inserted into the tree in a sorted manner using the InsertData() function. If the IC number is smaller, it is placed on the left side of the tree, and if it is larger, it goes to the right. This structure allows the system to search for a student very quickly.



```
void Insert(TreeNode*& tree, Activity data) // insert based on ic
{   if (tree == NULL) {
        tree = new TreeNode;
        tree->right = NULL;
        tree->left = NULL;
        tree->info = data;
}
    else if (data.ic < tree->info.ic)
        Insert(tree->left, data);
    else
        Insert(tree->right, data);
}

void Tree::InsertData(Activity data)
{
Insert(root, data);}
```

b) DeleteData function

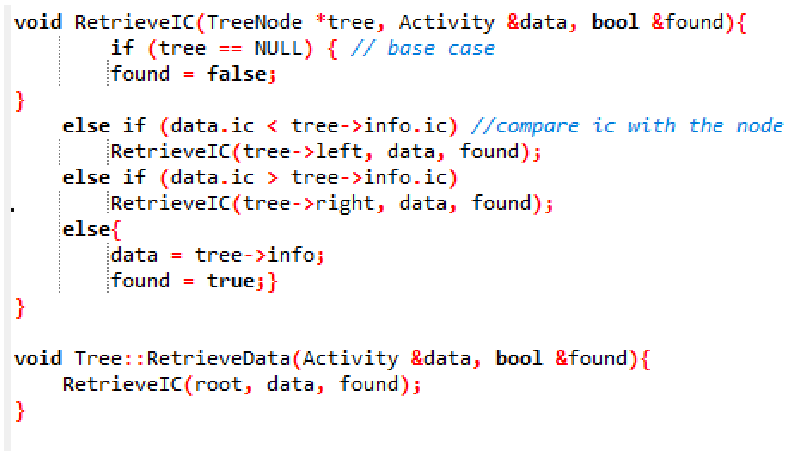
When a student is removed using the queue, the system also deletes the same student from the BST using the DeleteData() function. This ensures that both data structures contain the same updated information.



```
void Delete(TreeNode*& tree, string ic){
        if(tree == NULL)
            return;
        else if (ic < tree->info.ic)
            Delete(tree->left, ic);
        else if (ic > tree->info.ic)
            Delete(tree->right, ic);
        else{
            //one children
            if (tree->left == NULL){
                TreeNode *temp = tree;
                tree = tree->right;
                delete temp;
            }
            else if (tree->right == NULL){
                TreeNode *temp = tree;
                tree = tree->left;
                delete temp;
            }
            //two children
            else{
                TreeNode *temp = tree->right;
                while(temp->left != NULL){
                    temp = temp->left;
                }
                tree->info = temp->info;
                Delete(tree->right, temp->info.ic);

            }
        }

}

void Tree::DeleteData(string ic){
    Delete(root, ic);
}
```

c) RetrieveData function

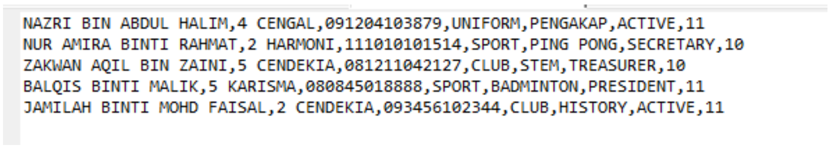
When the user searches for a student using their IC number, the system uses the RetrieveData() function to traverse the tree and find the matching record efficiently instead of checking every record one by one.


```
void RetrieveIC(TreeNode *tree, Activity &data, bool &found){
        if (tree == NULL) { // base case
        found = false;
}
    else if (data.ic < tree->info.ic) //compare ic with the node
        RetrieveIC(tree->left, data, found);
    else if (data.ic > tree->info.ic)
        RetrieveIC(tree->right, data, found);
    else{
        data = tree->info;
        found = true;}
}

void Tree::RetrieveData(Activity &data, bool &found){
    RetrieveIC(root, data, found);
}
```

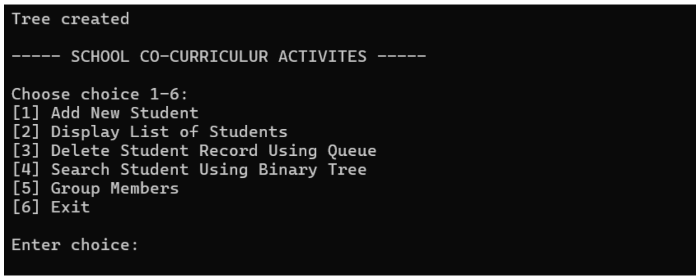
In conclusion, by combining the queue and the binary search tree, the system is able to support orderly deletion, fast searching, and efficient record management, making it suitable for handling school co-curricular data.

6.0 User Manual

The user opens the file curricular.txt and runs the miniprojectDSA.cpp program.



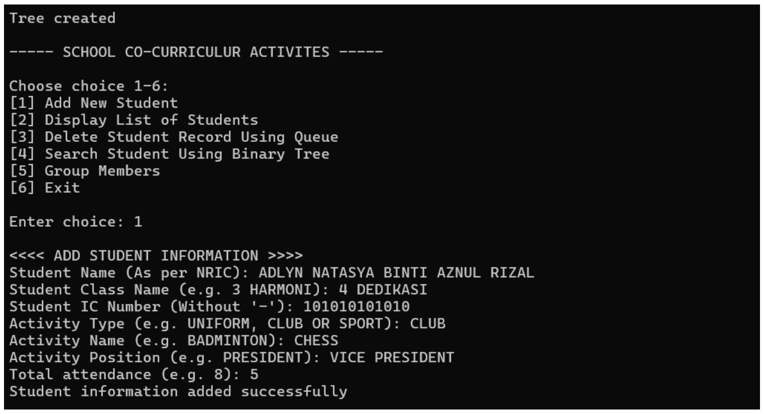
```
NAZRI BIN ABDUL HALIM,4 CENGAL,091204103879,UNIFORM,PENGAKAP,ACTIVE,11
NUR AMIRA BINTI RAHMAT,2 HARMONI,111010101514,SPORT,PING PONG,SECRETARY,10
ZAKWAN AQIL BIN ZAINI,5 CENDEKIA,081211042127,CLUB,STEM,TREASURER,10
BALQIS BINTI MALIK,5 KARISMA,080845018888,SPORT,BADMINTON,PRESIDENT,11
JAMILAH BINTI MOHD FAISAL,2 CENDEKIA,093456102344,CLUB,HISTORY,ACTIVE,11
```



```
Tree created

----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice:
```

Choice 1: Add New Student

Users need to enter student name, student class name, student IC number, activity type, activity name, activity position and total attendance. After that, the program displays “Student information added successfully” to show the student record already kept in the file “curriculur.txt”.

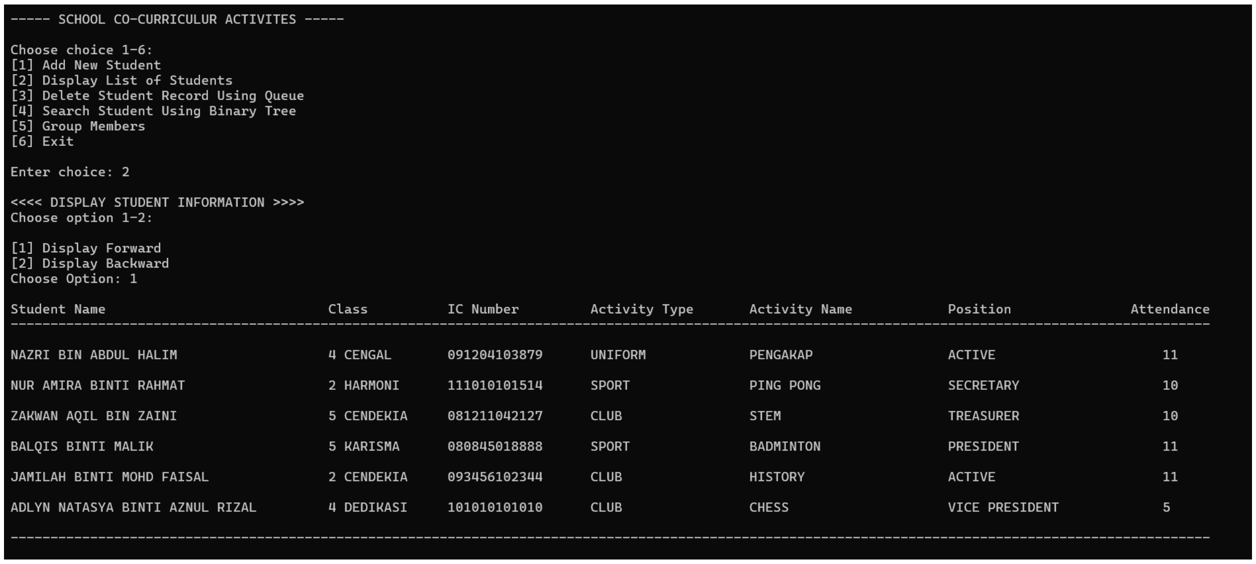


```
Tree created

----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 1

<<<< ADD STUDENT INFORMATION >>>>
Student Name (As per NRIC): ADLYN NATASYA BINTI AZNUL RIZAL
Student Class Name (e.g. 3 HARMONI): 4 DEDIKASI
Student IC Number (Without '-'): 101010101010
Activity Type (e.g. UNIFORM, CLUB OR SPORT): CLUB
Activity Name (e.g. BADMINTON): CHESS
Activity Position (e.g. PRESIDENT): VICE PRESIDENT
Total attendance (e.g. 8): 5
Student information added successfully
```

Choice 2: Display List of Students

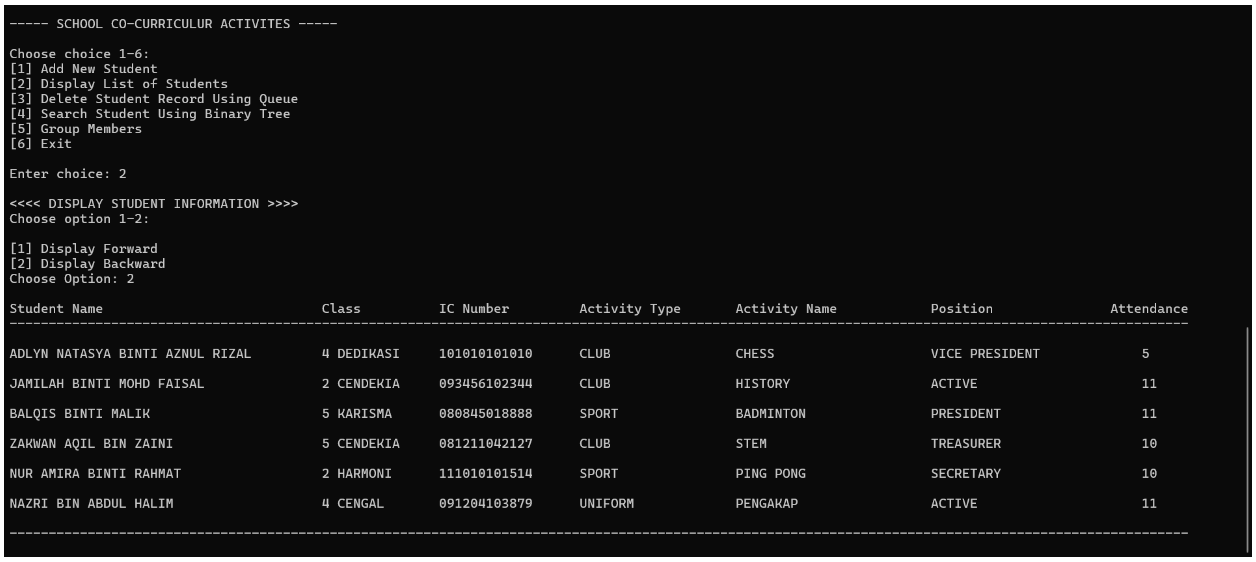
Users can choose two options of display which are [1] Display Forward and [2] Display Backward. The program displays the student name, student class name, student IC number, activity type, activity name, activity position and total attendance from the “curriculur.txt” file.

When the user chooses option 1, the program displays starting from the first student until the last student in the list.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 2

<<<< DISPLAY STUDENT INFORMATION >>>>
Choose option 1-2:

[1] Display Forward
[2] Display Backward
Choose Option: 1

Student Name                            Class          IC Number         Activity Type       Activity Name            Position              Attendance
-------------------------------------------------------------------------------------------------------------------------------------------------------

NAZRI BIN ABDUL HALIM                   4 CENGAL       091204103879      UNIFORM             PENGAKAP                 ACTIVE                    11

NUR AMIRA BINTI RAHMAT                  2 HARMONI      111010101514      SPORT               PING PONG                SECRETARY                 10

ZAKWAN AQIL BIN ZAINI                   5 CENDEKIA     081211042127      CLUB                STEM                     TREASURER                 10

BALQIS BINTI MALIK                      5 KARISMA      080845018888      SPORT               BADMINTON                PRESIDENT                 11

JAMILAH BINTI MOHD FAISAL               2 CENDEKIA     093456102344      CLUB                HISTORY                  ACTIVE                    11

ADLYN NATASYA BINTI AZNUL RIZAL         4 DEDIKASI     101010101010      CLUB                CHESS                    VICE PRESIDENT            5

-------------------------------------------------------------------------------------------------------------------------------------------------------
```

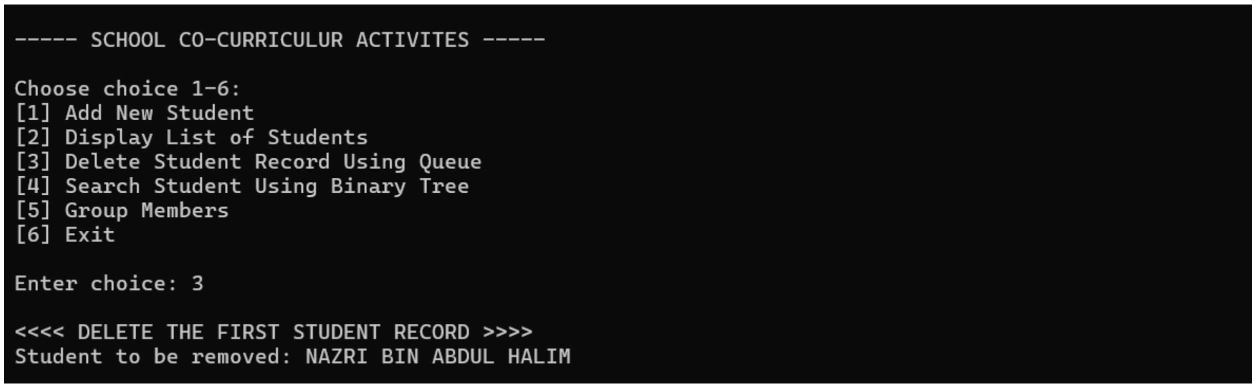
When the user chooses option 2, the program displays starting from the last student until the first student in the list.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 2

<<<< DISPLAY STUDENT INFORMATION >>>>
Choose option 1-2:

[1] Display Forward
[2] Display Backward
Choose Option: 2

Student Name                        Class         IC Number        Activity Type      Activity Name          Position             Attendance
--------------------------------------------------------------------------------------------------------------------------------------------

ADLYN NATASYA BINTI AZNUL RIZAL     4 DEDIKASI    101010101010     CLUB               CHESS                  VICE PRESIDENT           5

JAMILAH BINTI MOHD FAISAL           2 CENDEKIA    093456102344     CLUB               HISTORY                ACTIVE                   11

BALQIS BINTI MALIK                  5 KARISMA     080845018888     SPORT              BADMINTON              PRESIDENT                11

ZAKWAN AQIL BIN ZAINI               5 CENDEKIA    081211042127     CLUB               STEM                   TREASURER                10

NUR AMIRA BINTI RAHMAT              2 HARMONI     111010101514     SPORT              PING PONG              SECRETARY                10

NAZRI BIN ABDUL HALIM               4 CENGAL      091204103879     UNIFORM            PENGAKAP               ACTIVE                   11

--------------------------------------------------------------------------------------------------------------------------------------------
```

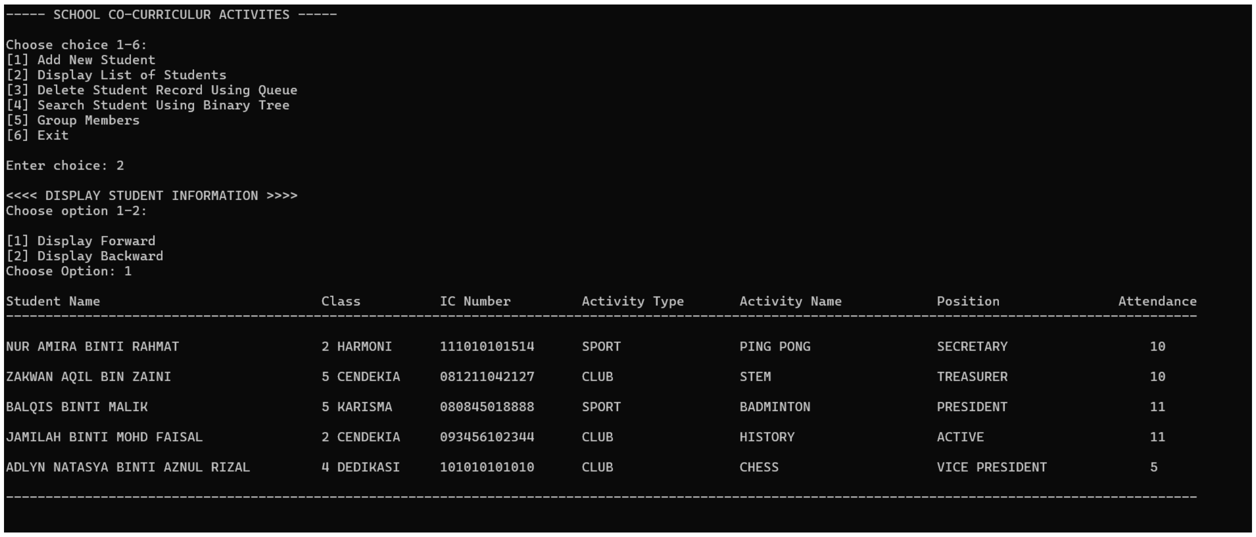
Choice 3: Delete Student Record Using Queue

When the user chooses choice 3, the first student record in the list will be deleted.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 3

<<<< DELETE THE FIRST STUDENT RECORD >>>>
Student to be removed: NAZRI BIN ABDUL HALIM
```

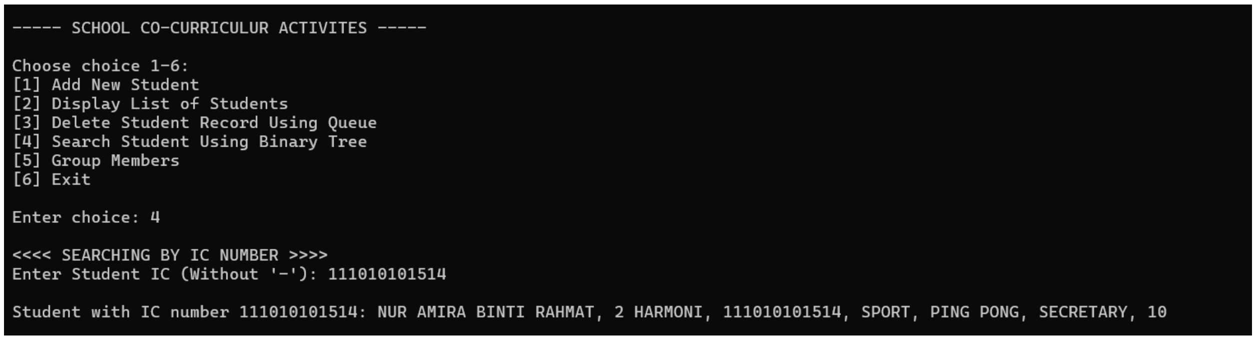
Example using [2] Display Forward after student NAZRI BIN ABDUL HALIM be removed.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 2

<<<< DISPLAY STUDENT INFORMATION >>>>
Choose option 1-2:

[1] Display Forward
[2] Display Backward
Choose Option: 1

Student Name                            Class          IC Number         Activity Type       Activity Name            Position               Attendance
-------------------------------------------------------------------------------------------------------------------------------------------------------

NUR AMIRA BINTI RAHMAT                  2 HARMONI      111010101514      SPORT               PING PONG                SECRETARY                  10

ZAKWAN AQIL BIN ZAINI                   5 CENDEKIA     081211042127      CLUB                STEM                     TREASURER                  10

BALQIS BINTI MALIK                      5 KARISMA      080845018888      SPORT               BADMINTON                PRESIDENT                  11

JAMILAH BINTI MOHD FAISAL               2 CENDEKIA     093456102344      CLUB                HISTORY                  ACTIVE                     11

ADLYN NATASYA BINTI AZNUL RIZAL         4 DEDIKASI     101010101010      CLUB                CHESS                    VICE PRESIDENT             5

-------------------------------------------------------------------------------------------------------------------------------------------------------
```

Choice 4: Search Student Using Binary Tree

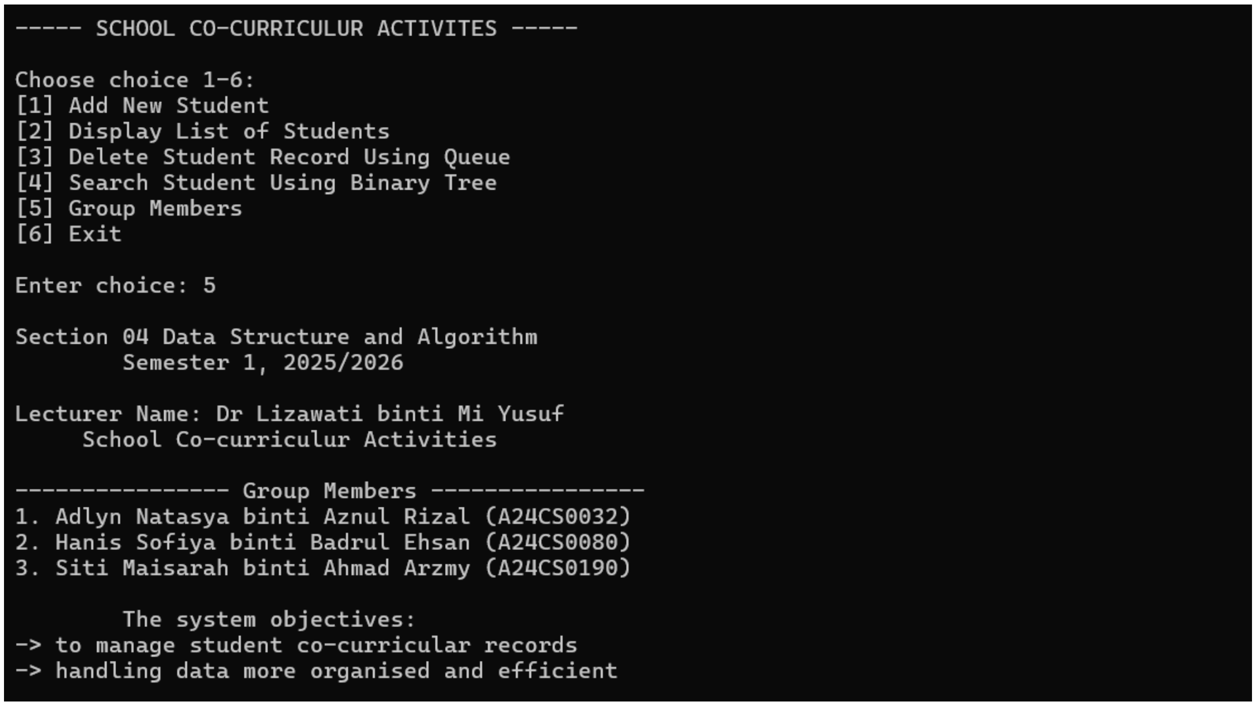
Users can find specific students using student IC. The program displays the student information if the student IC inserted matches with any student IC in the record.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 4

<<<< SEARCHING BY IC NUMBER >>>>
Enter Student IC (Without '-'): 111010101514

Student with IC number 111010101514: NUR AMIRA BINTI RAHMAT, 2 HARMONI, 111010101514, SPORT, PING PONG, SECRETARY, 10
```

Choice 5: Group Members

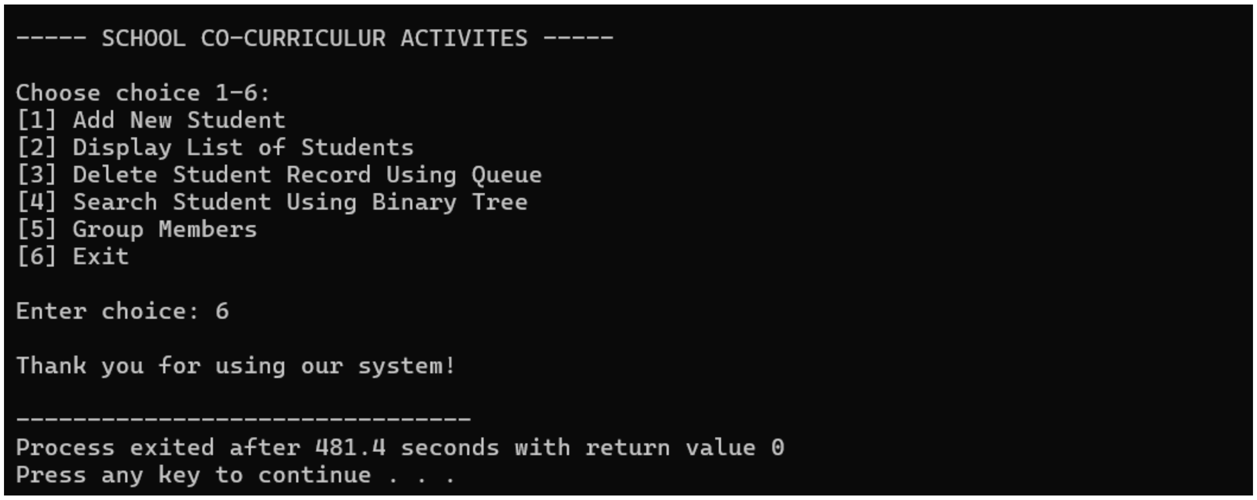
The program displays the information about the group members and the system objectives.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 5

Section 04 Data Structure and Algorithm
        Semester 1, 2025/2026

Lecturer Name: Dr Lizawati binti Mi Yusuf
     School Co-curriculur Activities

---------------- Group Members ----------------
1. Adlyn Natasya binti Aznul Rizal (A24CS0032)
2. Hanis Sofiya binti Badrul Ehsan (A24CS0080)
3. Siti Maisarah binti Ahmad Arzmy (A24CS0190)

        The system objectives:
-> to manage student co-curricular records
-> handling data more organised and efficient
```

Choice 6: Exit

When the user enters choice 6, the program displays "Thank you for using our system" and terminates.



```
----- SCHOOL CO-CURRICULUR ACTIVITES -----

Choose choice 1-6:
[1] Add New Student
[2] Display List of Students
[3] Delete Student Record Using Queue
[4] Search Student Using Binary Tree
[5] Group Members
[6] Exit

Enter choice: 6

Thank you for using our system!

--------------------------------
Process exited after 481.4 seconds with return value 0
Press any key to continue . . .
```

7.0 Source code

```
#include <iostream>
#include <string>
#include <fstream>
#include <iomanip>
using namespace std;
#define N 100

class Activity{
	public:
		string name;
		string className;
		string ic;
		string actName;
		string actType;
		string position;
		int attendance;
};


Activity a[N];
int totaldata = 0;

struct TreeNode{
	Activity info;
	TreeNode *left;
	TreeNode *right;
};

class Tree {
	public:
		Tree();
		bool IsEmpty()const;
		void RetrieveData(Activity&,bool& found);
		void InsertData(Activity);
		void DeleteData(string);
	private:
		TreeNode * root;
};

class Queue{
		Activity items[N];
		int front, back, count;

	public:
		Queue();
		bool isEmpty();
		bool isFull();
		void enQueue(Activity);
		void deQueue(Tree &);
		Activity getFront();
		Activity getBack();
		void dispForward();
		void dispBackward();
};



Tree::Tree(){
	root = NULL;
	cout << "Tree created" << endl;
```

```
}

bool Tree::IsEmpty() const
{       if (root == NULL)
                return true;
        else
                return false;
}


void Insert(TreeNode*& tree, Activity data)
{       if (tree == NULL) {
                tree = new TreeNode;
                tree->right = NULL;
                tree->left = NULL;
                tree->info = data;
}
        else if (data.ic < tree->info.ic)
                Insert(tree->left, data);
        else
                Insert(tree->right, data);
}

void Tree::InsertData(Activity data)
{
Insert(root, data);}

void RetrieveIC(TreeNode *tree, Activity &data, bool &found){
                if (tree == NULL) { // base case
                found = false;
}
```

```
        else if (data.ic < tree->info.ic)
                RetrieveIC(tree->left, data, found);
        else if (data.ic > tree->info.ic)
                RetrieveIC(tree->right, data, found);
        else{
                data = tree->info;
                found = true;}
}

void Tree::RetrieveData(Activity &data, bool &found){
        RetrieveIC(root, data, found);
}



void searchIC(Tree &s){
        string enter_ic;
        cout << "Enter Student IC (Without '-'): ";
        cin >> enter_ic;

        bool found = false;
        Activity temp;
        temp.ic = enter_ic;
        s.RetrieveData(temp, found);



                if(found){
                                cout << "\nStudent with IC number " << temp.ic << ": ";

                                cout << temp.name << ", "
                                        << temp.className << ", "
                                        << temp.ic << ", "
```

```
                                    << temp.actType << ", "
                                    << temp.actName << ", "
                                    << temp.position << ", "
                                    << temp.attendance << endl;

                }


            else // if the IC number not found
                cout << "Sorry, the IC number inserted does not exist.\n" << endl;
}

void Delete(TreeNode*& tree, string ic){
                if(tree == NULL)
                        return;
                else if (ic < tree->info.ic)
                        Delete(tree->left, ic);
                else if (ic > tree->info.ic)
                        Delete(tree->right, ic);
                else{
                                if (tree->left == NULL){
                                TreeNode *temp = tree;
                                tree = tree->right;
                                delete temp;
                        }
                        else if (tree->right == NULL){
                                TreeNode *temp = tree;
                                tree = tree->left;
                                delete temp;
                        }
                        else{
                                TreeNode *temp = tree->right;
                                while(temp->left != NULL){
                                        temp = temp->left;
                                }
                                tree->info = temp->info;
                                Delete(tree->right, temp->info.ic);

                        }
                }

}

void Tree::DeleteData(string ic){
        Delete(root, ic);
}



void readInput(){
        ifstream inputfile("curriculur.txt");

        if(!inputfile){
          cout << "Error! Input file cannot be opened." << endl;
          exit(1);
        }
        totaldata=0;

        while (getline(inputfile, a[totaldata].name, ',')){
                getline(inputfile, a[totaldata].className, ',');
                getline(inputfile, a[totaldata].ic, ',');
                getline(inputfile, a[totaldata].actType, ',');
                getline(inputfile, a[totaldata].actName, ',');
                getline(inputfile, a[totaldata].position, ',');

                inputfile >> a[totaldata].attendance;
                inputfile.ignore();  // ignore keyboard buffer

                totaldata++;
   }
        inputfile.close(); // close input file

}

void keepOutput(){ // to store student record
        ofstream outfile("curriculur.txt");

        for(int i = 0; i<totaldata; i++){
                outfile << a[i].name << ","
                                << a[i].className << ","
                                << a[i].ic << ","
                                << a[i].actType << ",";
                outfile << a[i].actName << ","
                                << a[i].position << ","
                                << a[i].attendance << endl;
        }

        outfile.close(); // close output file

}

void displayData(Queue &q){ // display student record
        if(totaldata == 0){
```

```
        cout << "Student list is empty." << endl;
        return;
}

int option;

do{
        cout << "Choose option 1-2:" << endl;
        cout << "\n[1] Display Forward" << endl;
        cout << "[2] Display Backward" << endl;
        cout << "Choose Option: ";
        cin >> option;

        if(option != 1 && option != 2){
                cout << "Invalid option. Please choose option again" << endl;
        }

} while (option != 1 && option != 2);

        cout << endl;
        cout << left
                << setw(40) << "Student Name"
                << setw(15) << "Class"
                << setw(18) << "IC Number"
                << setw(18) << "Activity Type";
        cout << right
                << setw(15) << "Activity Name"
                << setw(20) << "Position"
                << setw(25) << "Attendance" << endl;
```

```
                    cout <<
"-------------------------------------------------------------------------------------------------------------------
-----------------------------------" << endl << endl;

                    if(option == 1){
                              q.dispForward();
                    }
                    else if(option == 2){
                              q.dispBackward();
                    }
                    else{
                              cout << "Invalid option" << endl;
                    }



  cout <<
"-------------------------------------------------------------------------------------------------------------------
-----------------------------------" << endl;
  cout << endl;
}



void addStudent(Tree &s, Queue &q){ // add student to the list
          if (totaldata >= N){
                    cout << "Sorry, the Student list already full!" << endl;
                    return;
          }

          cin.ignore();
          cout << "Student Name (As per NRIC): ";
```

```
        getline(cin, a[totaldata].name);
        cout << "Student Class Name (e.g. 3 HARMONI): ";
        getline(cin, a[totaldata].className);
        cout << "Student IC Number (Without '-'): ";
        getline(cin, a[totaldata].ic);
        cout << "Activity Type (e.g. UNIFORM, CLUB OR SPORT): ";
        getline(cin, a[totaldata].actType);
        cout << "Activity Name (e.g. BADMINTON): ";
        getline(cin, a[totaldata].actName);
        cout << "Activity Position (e.g. PRESIDENT): ";
        getline(cin, a[totaldata].position);
        cout << "Total attendance (e.g. 8): ";
        cin >> a[totaldata].attendance;


        totaldata++;
        keepOutput();
        q.enQueue(a[totaldata - 1]);
        s.InsertData(a[totaldata - 1]);
        cout << "Student information added successfully" << endl;
}



Queue::Queue(){
        count = 0;
        front = 0;
        back = N - 1;
}
```

```
bool Queue::isEmpty(){
        return (count == 0);
}

bool Queue::isFull(){
        return (count == N);
}

void Queue::enQueue(Activity n){
        if (isFull())
                cout << "Sorry, the queue is full" << endl;
        else{
                back = (back + 1) % N;
                items[back] = n;
                count++;
        }
}

void Queue::deQueue(Tree &s){
        if(isEmpty())
                cout << "Sorry, the queue is empty" << endl;
        else{
                cout << "Student to be removed: " << items[front].name << endl;

                s.DeleteData(items[front].ic);

        for(int i = 0; i < totaldata - 1; i++){
                a[i] = a[i + 1];
                }

                totaldata--;
                front = (front + 1) % N;
                --count;
                keepOutput();

                }
}

Activity Queue::getFront(){
        return items[front];
}

Activity Queue::getBack(){
        return items[back];
}

void Queue::dispForward(){
        for(int i = front; i <= back; i++){
                cout << left
                        << setw(40) << items[i].name
                        << setw(15) << items[i].className
                        << setw(18) << items[i].ic
                        << setw(20) << items[i].actType
                        << setw(25) << items[i].actName
                        << setw(27) << items[i].position
                        << setw(5) << items[i].attendance << endl << endl;
        }

        }

void Queue::dispBackward(){
        for(int i = back; i >= front; i--){
```

```
			cout << left
				<< setw(40) << items[i].name
				<< setw(15) << items[i].className
				<< setw(18) << items[i].ic
				<< setw(20) << items[i].actType
				<< setw(25) << items[i].actName
				<< setw(27) << items[i].position
				<< setw(5) << items[i].attendance << endl << endl;
		}

			}



int main(){
		readInput();
		Queue q;
		Tree s;

		for(int i = 0; i < totaldata; i++){
			q.enQueue(a[i]);
			s.InsertData(a[i]);
		}

		int choice;

		do{
		cout << "\n----- SCHOOL CO-CURRICULUR ACTIVITES -----" << endl << endl;
		cout << "Choose choice 1-6:" << endl;
		cout << "[1] Add New Student" << endl
			 << "[2] Display List of Students" << endl
			 << "[3] Delete Student Record Using Queue" << endl
```

```
                << "[4] Search Student Using Binary Tree" << endl
                << "[5] Group Members" << endl
                << "[6] Exit" << endl;



        cout << "\nEnter choice: ";
        cin >> choice;
        cout << endl;

        switch(choice){
                case 1: cout << "<<<< ADD STUDENT INFORMATION >>>>" << endl;
                                addStudent(s, q);
                                break;

                case 2: cout << "<<<< DISPLAY STUDENT INFORMATION >>>>" << endl;
                                displayData(q);
                                break;

                case 3: cout << "<<<< DELETE THE FIRST STUDENT RECORD >>>>" <<
endl;
                                q.deQueue(s);
                                break;

                case 4:  cout << "<<<< SEARCHING BY IC NUMBER >>>>" << endl;
                                 searchIC(s);
                                 break;

                case 5: cout << "Section 04 Data Structure and Algorithm" << endl
                                        << "\tSemester 1, 2025/2026" << endl << endl
                                        << "Lecturer Name: Dr Lizawati binti Mi Yusuf" << endl
                                        << "     School Co-curriculur Activities" << endl << endl
```

```
                                                  << "---------------- Group Members ----------------" << endl
                                                  << "1. Adlyn Natasya binti Aznul Rizal (A24CS0032)" <<
endl
                                                  << "2. Hanis Sofiya binti Badrul Ehsan (A24CS0080)" <<
endl
                                                  << "3. Siti Maisarah binti Ahmad Arzmy (A24CS0190)"
<< endl << endl
                                                  << "\tThe system objectives:" << endl
                                                  << "-> to manage student co-curricular records" << endl
                                                  << "-> handling data more organised and efficient" <<
endl << endl;
                                          break;

                    case 6: cout << "Thank you for using our system!" << endl;
                                          return 0; // program terminates

                    default: cout << "Invalid choice. Please try again" << endl;
          }} while (choice != 6);

          return 0;
}
```