```
#include<iostream>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <string>
using namespace std;

struct Date {
	int Day, Month, Yeah;
};

struct Student {
	char ID[50];
	char Name[50];
	char Country[50];
	Date Birth;
	double Math;
	double Physical;
	double Chemistry;
};

struct Node
{
	Student Key;
	struct Node *Left;
	struct Node *Right;
};
typedef struct Node NODE;
typedef NODE* TREE;

int compare(Student x, Student y)
{
	return strcmp(x.ID, y.ID);
}

int compare(char x[50], char ID[50])
{
	return strcmp(x, ID);
}

Student Input()
{
	Student x;
	cout << "Add ID (Q to return): ";
	gets(x.ID);
	if (strcmp(x.ID, "q") == 0 || strcmp(x.ID, "Q") == 0)
```

```
{
	return x;
}
cout << "Add name: ";
gets(x.Name);
cout << "Add country: ";
gets(x.Country);
cout << "Add Day/ Month/ Yeah: \n";
cout << "Add Day: ";
cin >> x.Birth.Day;
cout << "Add Month: ";
cin >> x.Birth.Month;
cout << "Add Yeah: ";
cin >> x.Birth.Yeah;
cout << "Add Math Score: ";
while (true)
{
	cin >> x.Math;
	if (cin.fail() || x.Math > 10 || x.Math < 0)
	{
		cin.clear();
		_flushall();
		cin.ignore();
		cout << "Add Math Score Again: ";
	}
	else
	{
		break;
	}
}
cout << "Add Physical Score: ";
while (true)
{
	cin >> x.Physical;
	if (cin.fail() || x.Physical > 10 || x.Physical < 0)
	{
		cin.clear();
		_flushall();
		cin.ignore();
		cout << "Add Physical Score Again: ";
	}
	else
	{
		break;
	}
}
```

```
	cout << "Add Chemistry Score: ";
	while (true)
	{
		cin >> x.Chemistry;
		if (cin.fail() || x.Chemistry > 10 || x.Chemistry < 0)
		{
			cin.clear();
			_flushall();
			cin.ignore();
			cout << "Add Physical Score Again: ";
		}
		else
		{
			break;
		}
	}
	while (getchar() != '\n');
	return x;
}

void Output(Student x)
{
	cout << "========================================" << endl;
	cout << "ID: " << x.ID << "\n";
	cout << "Student Name: " << x.Name << "\n";
	cout << "Date: " << x.Birth.Day << "/" << x.Birth.Month << "/" << x.Birth.Yeah << "\n";
	cout << "Country: " << x.Country << "\n";
	cout << "Math Score: " << x.Math << "\n";
	cout << "Physical Score: " << x.Physical << "\n";
	cout << "Chemistry Score: " << x.Chemistry << "\n";
}

int InsertNode(TREE &t, Student x)
{
	if (t != NULL)
	{
		if (compare(t->Key, x) == 0)
		{
			return -1;
		}
		if (compare(t->Key, x) > 0)
		{
			return InsertNode(t->Left, x);
		}
		if (compare(t->Key, x) < 0)
		{
```

```
                return InsertNode(t->Right, x);
            }
        }
        t = (NODE*)malloc(sizeof(NODE));
        if (t == NULL)
        {
            cout << "Not enough memory!";
            return 0;
        }
        t->Key = x;
        t->Left = t->Right = NULL;
        return 1;
}

void CreateTree(TREE &t)
{
        Student x;
        while (1)
        {
            cout << "--Add student your information--" << endl;
            x = Input();
            if (strcmp(x.ID, "q") == 0 || strcmp(x.ID, "Q") == 0)
            {
                break;
            }
            int check = InsertNode(t, x);
            if (check == -1)
            {
                cout << "---ID Available---\n" << endl;
            }
            else if (check == 0)
            {
                cout << "---Full Memory---\n" << endl;
            }
            else
                cout << "---Add Success---\n\n";
        }
}

void LNR(TREE T)
{
        if (T != NULL)
        {
            LNR(T->Left);
            Output(T->Key);
            LNR(T->Right);
```

```
	}
}

NODE* Search(TREE t, char ID[50])
{
	Student x;
	if (t == NULL)
	{
		return NULL;
	}
	else
	{
		if (compare(t->Key.ID, ID) == 0)
		{
			return t;
		}
		else if(compare(t->Key.ID, ID) > 0)
		{
			Search(t->Left, ID);
		}
		else if(compare(t->Key.ID, ID) < 0)
		{
			Search(t->Right, ID);
		}
	}

}

void NodeReplace(TREE &X, TREE &Y)
{
	if (Y->Right != NULL)
	{
		NodeReplace(X, Y->Right);
	}
	else
	{
		X->Key = Y->Key;
		X = Y;
		Y = Y->Left;
	}
}

void LookingForNodeReplace(TREE &X, TREE &Y)
{
	if (Y->Left != NULL)
	{
```

```
                LookingForNodeReplace(X, Y->Left);
        }
        else
        {
                X->Key = Y->Key;
                X = Y;
                Y = Y->Right;
        }
}

NODE* DeleteNode(TREE &t, char ID[50])
{
        if (t == NULL)
        {
                return NULL;
        }
        else if(t != NULL)
        {
                if (compare(t->Key.ID, ID) == 0)
                {
                        return t;
                }
                else if(compare(t->Key.ID, ID) > 0)
                {
                        DeleteNode(t->Left, ID);
                }
                else if(compare(t->Key.ID, ID) < 0)
                {
                        DeleteNode(t->Right, ID);
                }
                else
                {
                        NODE *X = t;
                        if (t->Left == NULL)
                        {
                                t = t->Right;
                        }
                        else if (t->Right == NULL)
                        {
                                t = t->Left;
                        }
                        else
                        {
                                LookingForNodeReplace(X, t->Right);
                        }
                        delete X;
```

```
		}
	}
}


void Menu(TREE &t)
{
	int Option;
	while (true)
	{
		system("cls");
		cout << "*********************************************";
		cout << "\n* 1. Add information student              *";
		cout << "\n* 2. Show Students                        *";
		cout << "\n* 3. Search Student                       *";
		cout << "\n* 4. Delete                               *";
		cout << "\n* 5. Exit                                 *";
		cout << "\n*********************************************";
		cout << "\n\nChoose Option ?(1->5)";
		cin >> Option;

		if (Option == 1)
		{
			Student x;
			gets(x.ID);
			CreateTree(t);
		}
		else if (Option == 2)
		{
			cout << "\n\t BINARY SEARCH TREE\n";
			LNR(t);
			system("pause");
		}
		else if (Option == 3)
		{
			char ID[50];
			cout << "\nEnter ID for Search: ";
			fflush(stdin);
			gets(ID);
			NODE *p = Search(t, ID);
			if (p == NULL)
			{
				cout << "\nID " << ID << " not available!\n";
			}
			else
			{
```

```
                                cout << "\nID " << ID << " available!\n";
                                cout << "Information: \n";
                                cout <<"ID: " << p->Key.ID << "\n";
                                cout <<"Name: "<<p->Key.Name << "\n";
                                cout <<"Country: "<<p->Key.Country << "\n";
                                cout << "Date: " << p->Key.Birth.Day << "/" <<
p->Key.Birth.Month << "/" << p->Key.Birth.Yeah << "\n";
                                cout << "Math: " << p->Key.Math << "\n";
                                cout << "Physical: " << p->Key.Physical << "\n";
                                cout << "Chemistry: " << p->Key.Chemistry << endl;
                        }
                        system("pause");
                }
                else if (Option == 4)
                {
                        char ID[50];
                        cout << "\nEnter ID for Delete: ";
                        fflush(stdin);
                        gets(ID);
                        NODE *p = Search(t, ID);
                        if (p != NULL)
                        {
                                DeleteNode(t, ID);
                        }
                        else
                        {
                                cout << "Not";
                        }
                }
                else if (Option == 5)
                {
                        break;
                }
        }
}


int main()
{
        TREE t;
        t = NULL;
        Menu(t);
        system("pause");
        return 0;
}
```