Table of Content

Specific Requirement

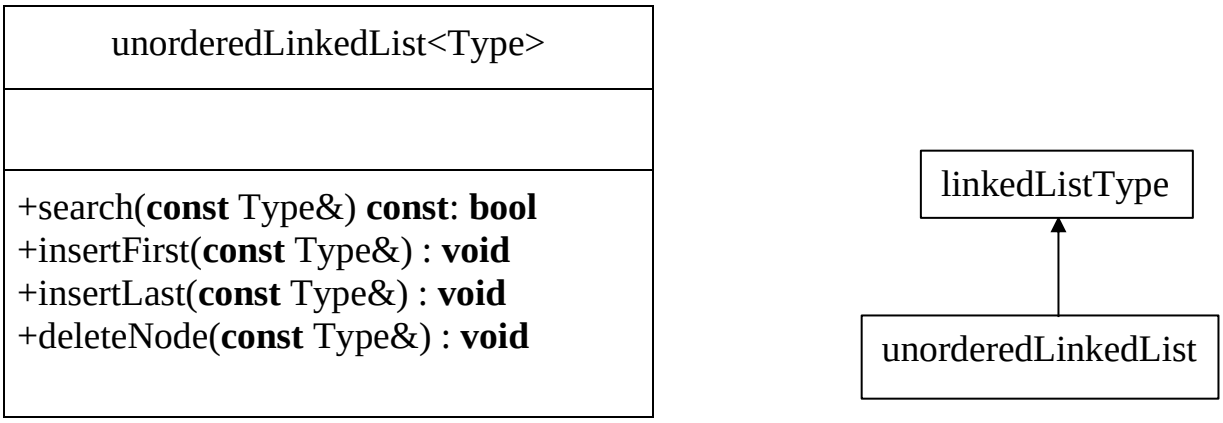
UML Diagram

linkedListIterator<Type>
-*current : nodeType<Type>
+linkedListIterator( ) +linkedListIterator(nodeType<Type>) +**operator***( ) : Type +**operator**++( ) : linkedListIterator<Type> +**operator**==(**const** linkedListIterator<Type>&) **const**: bool +**operator**!=(**const** linkedListIterator<Type>&) **const**: bool

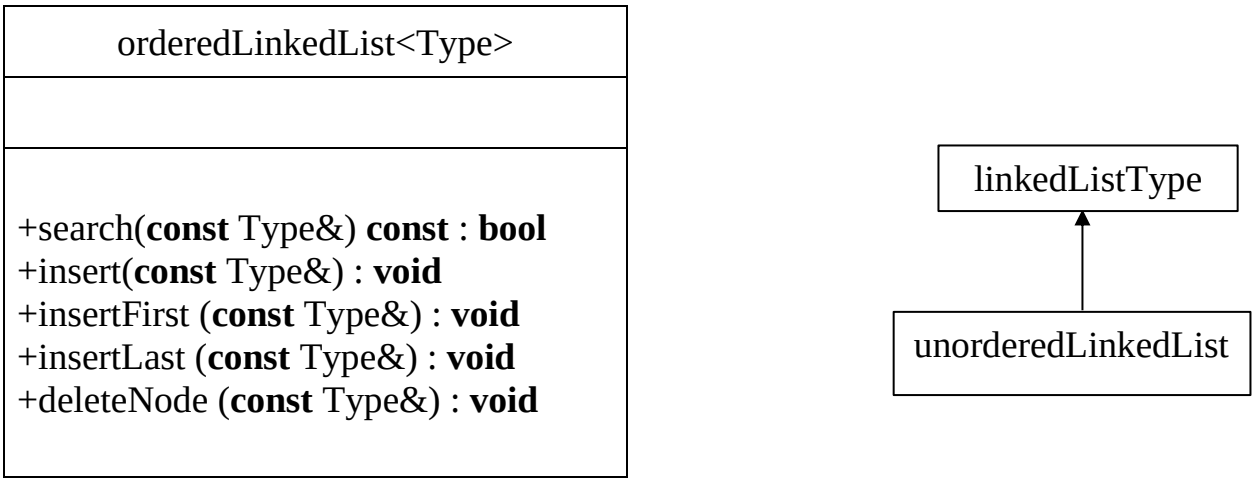
UML class diagram of the class linkedListIterator

linkedListType<Type>
#num : **int** #*first : nodeType<Type> #*last : nodeType<Type>
+**operator**= (**const** linkedListType<Type>&) : **const** linkedListType<Type>& +initializeList() : **void** +isEmptyList() **const**: **bool** +print() **const** : **void** +length() **const** : **int** +destroyList() : **void** +front() **const** : Type +back() **const** : Type +search(**const** Type&) **const** = 0 : **bool** +insertFirst(**const** Type&) = 0 : **void** +insertLast(**const** Type&) = 0 : **void** +deleteNode(**const** Type&) = 0 : **void** +begin () : linkedListIterator<Type> +end() : linkedListIterator<Type> +linkedListType() +linkedListType(**const** linkedListType<Type>&) +~linkedListType() -copyList(**const** linkedListType<Type>&) : **void**

UML class diagram of the class linkedListType



UML class diagram of the derived class unorderedLinkedList



UML class diagram of the derived class orderedLinkedList

Problem Analysis

Input	● Student name ● Student matric number ● Student CGPA ● Year of study
Process	1. Display main menu a. Prompt user to enter choice i. If user enters ‘1’, proceed to add student information ii. If user enters ‘2’, proceed to delete student information iii. If user enters ‘3’, proceed to search and display student information iv. If user enters ‘4’, proceed to display full student information list v. If user enters ‘5’, exit the program vi. If user enters other than 1~5, display error message and return to main menu 2. Add student information a. Prompt user to enter student name b. Prompt user to enter matric number c. Prompt user to enter CGPA i. If user enter a invalid CGPA (CGPA less than 0 or more than 4 or CGPA is not a value) 1. Display error message 2. Prompt user to re-enter d. Prompt user to enter year of study i. If user enter a invalid year of study (year less than 1 or more than 5 or year is not a value) 1. Display error message 2. Prompt user to re-enter e. Return to main menu 3. Delete student information

	a. Display existing student information b. Prompt user to enter matric number i. If the list is empty 1. Display error message 2. Return to main menu ii. If matric number not found 1. Display error message 2. Return to main menu iii. If matric number found 1. Return to main menu 4. Search and Display student information a. Prompt user to enter matric number i. If matric number not found 1. Display error message 2. Return to main menu ii. If matric number found 1. Display the relevant student information 2. Return to main menu 5. Exit the program
Output	● Student name ● Student matric number ● Student CGPA ● Year of study
Test Data	➔ Dolly, ll48y48, 3.98, 1 ➔ Nemo, 774nm, 2.65, 3 ➔ Bob, bob4524, 0.97, 5 ➔ Scooby, scb6666, 3.88, 4 ➔ Popeye, 1p2p3y, 1.23, 2

Design

Pseudocode

```
WHILE (isSystemOn)
START
Display main menu
Get user's choice

CASE 1 :
        Display add information
        Prompt user to enter Name
        Prompt user to enter Matric Number
        Prompt user to enter CGPA
                WHILE ( CGPA less than 0.00 or CGPA more than 4.00 )
                        PRINT error message
                        Prompt user to re-enter CGPA
                ENDWHILE
        Prompt user to enter Year of Study
                WHILE ( Year of Study less than 1 or Year of Study more than
                    4)
                        PRINT error message
                        Prompt user to re-enter Year of Study
                ENDWHILE
      Call function set_StudentInfo using object HomoSapien
        Passing the parameters and stored inside StudentInfo class variables
                store Name inside studentName
                store Matric Number inside studentMatric
                store CGPA inside studentCGPA
                store Year of Study inside studentYear
        Call unorderedLinkedList class function insertFirst using object UnorderedLL
        Object Homosapien is passed as parameter to the function and store in      newItem
                create newNode
                declare first and last pointer
                store newItem into newNode info
                assign first pointer points to newNode next
                make first point to the actual first node
                increment num
                IF (list is empty)
                        newNode is the last node in the list
                ENDIF
        ENDINSERTFIRST

        Call orderedLinkedList class function insert by using object OrderedLL
        Pass Object Homosapien  as parameter to the function and store in  newItem
                create newNode
                declare current and trail current pointer
                store newItem into newNode info
```

```
                    set the link field of the node to NULL
                    IF (list is initially empty)
                        first and last points to the new list
                        newItem is inserted at the proper place in the list
                        increment num
                    ELSE
                        start searching at first node using current pointer
                        found = false
                        WHILE ( current not equal to NULL AND is true)
                            IF ( current info.get_StudentMatric larger or equal to newNode
info.get_StudentMatric)
                                found = true
                            ELSE
                                trailCurrent point to current
                                current point to current next and traverse the list
                                IF ( current and first point to the same)
                                    newNode next equals to first
                                    make first pointer to the actual first node
                                    increment num
                                ELSE ( current equals to NULL)
                                    last point to the newNode
                                increment num
                            ENDIF
                    ENDIF
            ENDINSERT
            BREAK

            CASE 2 :
                    Call orderedLinkedList class function print by using object OrderedLL
                    display student data list
                    PRINT delete message
                    Get user delete Matric Number
                    Call unorderedLinkedList class function deleteNode by using object UnorderedLL

                    Pass Matric Number  as parameter to the function and store in deleteItem
                    declare current, trailCurrentfirst, first and last pointer
                    If ( the list is empty)
                        PRINT error message for delete node
                    ELSE
                        IF ( first info.get_StudentMatric equals to deleteItem)
                            current point to first
                            first point to first next
                            decrement num
                            IF (list has only one node )
                                last point to NULL
                            DELETE current
                        ELSE
                            found = false
                            set trailCurrent points to first node
```

```
                    set current points to second node
                    WHILE ( current not equals to NULL AND is true)
                        IF ( current info.get_StudentMatric not equals to deleteItem
                            trailCurrent point to current
                            current point to current next and traverse the list
                        ELSE
                            found = true
                        ENDIF
                    ENDWHILE

                    IF ( found)
                        trail next equals to current next
                        decrement num
                        IF ( node to be deleted was the last node)
                            Update the value of last
                            DELETE the node from the list
                    ELSE
                        PRINT error message for delete node
                    ENDIF
                ENDIF
            ENDIF
        ENDDELETENODE

        Call orderedLinkedList class function deleteNode by using Matric Number

            Pass Matric Number  as parameter to the function and store in deleteItem
            declare current, trailCurrentfirst, first and last pointer
            If ( the list is empty)
                PRINT error message for delete node
            ELSE
                IF ( first info.get_StudentMatric equals to deleteItem)
                    current point to first
                    first point to first next
                    decrement num
                    IF (list has only one node )
                        last point to NULL
                    DELETE current
                ELSE
                    found = false
                    set trailCurrent points to first node
                    set current points to second node
                    WHILE ( current not equals to NULL AND is true)
                        IF ( current info.get_StudentMatric not equals to deleteItem
                            trailCurrent point to current
                            current point to current next and traverse the list
                        ELSE
                            found = true
                        ENDIF
                    ENDWHILE
```

```
                                IF ( found)
                                        trail next equals to current next
                                        decrement num
                                        IF ( node to be deleted was the last node)
                                                Update the value of last
                                                DELETE the node from the list
                                ELSE
                                        PRINT error message for delete  node
                                ENDIF
                        ENDIF
                ENDIF
        ENDDELETENODE

        CASE 3 :
                Get user search matric number
                Call orderedLinkedList class function search by using object OrderedLL
                Pass Matric Number  as parameter to the function and store in searchItem
                declare current
                Start the search at the first node
                WHILE (current not equal to NULL AND not found)
                        IF (current info.get_StudentMatric() equal to searchItem)
                                found = true
                        ELSE
                                current point to next
                ENDWHILE
                IF (found)
                        print found message
                ELSE
                        print error message

                Call unorderedLinkedList class function search by using object UnorderedLL
                Pass Matric Number  as parameter to the function and store in searchItem
                declare current
                Start the search at the first node
                WHILE (current not equal to NULL AND not found)
                        IF (current info.get_StudentMatric() equal to searchItem)
                                found = true
                        ELSE
                                current point to next
                ENDWHILE
                IF (found)
                        print found message
                ELSE
                        print error message
        ENDSEARCH

        CASE 4 :
                Call orderedLinkedList class function print by using object OrderedLL
```

```
            display student data list
            Call unorderedLinkedList class function print by using object UnorderedLL
            display student data list
        ENDPRINT

        CASE 5:
            print exit message
            EXIT

        DEFAULT:
            print error message
            return to menu
```

C++ Program List

Main

```
#include <iostream>
#include <string>
#include <iomanip>
#include <conio.h>
#include "Linkedlist.h"
#include "StudentInfo.h"

using namespace std;

void header();
void printMenu();
//Function to show menu

int main()
{
	StudentInfo HomoSapien;
	orderedLinkedList<StudentInfo> OrderedLL;
	unorderedLinkedList<StudentInfo> UnorderedLL;

	bool isSystemOn = true;
	int choice;
	string n,m;
	float c;
	int y;

	while(isSystemOn)
	{
		header();
		printMenu();
		cout << endl;
		cout << "\tChoice: ";
		cin >> choice;

		switch(choice)
		{
			case 1: //Add information
				{
					cin.clear();
					cin.ignore(1000,'\n');

					system("cls");
					header();
```

```
                    cout << "\tAdd Student Information: " << endl;
                    cout <<
"==========================================================================================
=======" << endl;
                    cout << "\n\t-> Name: ";
                    getline(cin, n);

                    cout << "\t-> Matric Number: ";
                    getline(cin, m);

                    cout << "\t-> CGPA: ";
                    cin >> c;
                    while ((c < 0.00) || (c>4.00) || cin.fail())
                    {
                              cin.clear();
                              cin.ignore(1000,'\n');
                              cout << "\t   Invalid CGPA..." << endl;
                              cout << "\t   Re-enter CGPA: ";
                              cin >> c;
                    }

                    cin.clear();
                    cin.ignore(1000,'\n');
                    cout << "\t-> Year of study: ";
                    cin >> y;
                    while ((y < 1) || (y>5) || cin.fail())
                    {
                         cin.clear();
                         cin.ignore(1000,'\n');
                         cout << "\t   Invalid Year of Study..." << endl;
                         cout << "\t   Re-enter Year of Study: ";
                         cin >> y;
                    }

                    HomoSapien.set_StudentInfo(n,m,c,y);

                    UnorderedLL.insertFirst(HomoSapien);
                    OrderedLL.insert(HomoSapien);
                    system("pause");
                    system("cls");
                    break;
          }

     case 2: //Delete information
          {
                    system("cls");
                    header();

                    cout << "\n";
```

```
                    OrderedLL.print(); //To display the student data list
                    cout << "\n";

                    cout << endl << endl << endl;
                    cout << "\tDelete Student Information: " << endl;
                    cout <<
"==================================================================================
=======" << endl;
                    cin.clear();
                    cin.ignore(1000,'\n');

                    cout << "\tMatric number: ";
                    getline(cin, m);

                    UnorderedLL.deleteNode(m);
                    OrderedLL.deleteNode(m);
                    system("pause");
                    system("cls");
                    break;
                }

            case 3: //Search information
                {
                    system("cls");
                    header();

                    cout << "\tSearch Student Information: " << endl;
                    cout <<
"==================================================================================
=======" << endl;

                    cin.clear();
                    cin.ignore(1000,'\n');

                    cout << "\tMatric number: ";
                    getline(cin, m);

                    cout << endl << endl << endl;
                    UnorderedLL.search(m);
                    cout << endl << endl << endl;

                    OrderedLL.search(m);
                    cout << endl << endl << endl;

                    system("pause");
                    system("cls");
                    break;
                }
```

```
                case 4: //Display Student Information
                        {
                                cin.clear();
                                cin.ignore(1000,'\n');
                                system("cls");
                                header();

                                cout << "\tDisplay Student Information: " << endl;
                                cout <<
"==================================================================================
=======" << endl;

                                cout << "\n\tPrint Student Information using Ordered Linked
List" << endl;
                                OrderedLL.print();

                                cout << "\n\n\tPrint Student Information using Unordered
Linked List" << endl;
                                UnorderedLL.print();
                                system("pause");
                                system("cls");
                                break;
                        }

                case 5:
                        {
                                system("cls");
                                header();
                                cout << "\t\t\t     < Closing the Program >" << endl << endl;
                                system("pause");
                                return 0;
                        }

                default:
                        {
                                system("cls");
                                header();
                                cout << "\tInvalid input..." << endl << endl;
                                cout << "\tPlease enter 1 to 5 ONLY" << endl << endl;
                                cout << "\tPress \"Enter\" to return to Menu..." << endl <<
endl;

                                cin.clear();
                                cin.ignore(1000,'\n');
                                system("pause");
                                system("cls");
                                break;
                        }
```

```
                }
        }
}

void header()
{
        cout << "\n\n\n";
        cout << "\t\t\tWELCOME TO UNIVERSITI SAINS MALAYSIA\n";
        cout << "\t\t\t    Student Information System\n\n";
        cout << endl << endl;
}

void printMenu()
{

        cout << "\tThis program is to store and organize students' information " << endl;
        cout <<
"==================================================================================
=======" << endl;
        cout << "\n       Student's Information: " << endl;
        cout << "\n        # Add ------------------------------------------------------- Enter '1'" << endl;
        cout << "\n        # Delete ----------------------------------------------------- Enter '2'" << endl;
        cout << "\n        # Search & Display ------------------------------------------- Enter '3'" << endl;
        cout << "\n        # Display Full Information List ------------------------------ Enter '4'" << endl;
        cout << "\n        # Exit Program ----------------------------------------------- Enter '5'" << endl;
        cout << endl;
        cout <<
"==================================================================================
=======" << endl;
}
```

StudentInfo.h

```
#ifndef Included_StudentInfo_H
#define Included_StudentInfo_H

#include<iostream>
#include<string>
#include<iomanip>

using namespace std;


class StudentInfo
{
   public:
         StudentInfo(){} //Constructor

         void set_StudentInfo(string, string, float, int);
         //Function to set all student info
         void set_StudentName(string);
         //Function to set student's name
         void set_StudentMatric(string);
         //Function to set student's matric number
         void set_Student_CGPA(float);
         //Function to set student's CGPA
         void set_StudentYear(int);
         //Function to set student's year of study

         //void get_StudentInfo(string, string, float, int);
         //Function to return all student info
         string get_StudentName();
         //Function to return student's name
         string get_StudentMatric();
         //Function to return student's matric number
         float get_Student_CGPA();
         //Function to return student's CGPA
         int get_StudentYear();
         //Function to return student's year of study

         void print_StudentInfo() const;
         //Display student info

         ~StudentInfo(); //Destructor;

   private:
                  string studentName;
                  string studentMatric;
                  float studentCGPA;
                  int studentYear;
```

```
};

void StudentInfo::set_StudentInfo(string n, string m, float c, int y)
{
	studentName = n;
	studentMatric = m;
	studentCGPA = c;
	studentYear = y;
}

void StudentInfo::set_StudentName(string n)
{
	studentName = n;
}

void StudentInfo::set_StudentMatric(string m)
{
	studentMatric = m;
}

void StudentInfo::set_Student_CGPA(float c)
{
	studentCGPA = c;
}

void StudentInfo::set_StudentYear(int y)
{
	studentYear = y;
}

string StudentInfo::get_StudentName()
{
	return studentName;
}

string StudentInfo::get_StudentMatric()
{
	return studentMatric;
}

float StudentInfo::get_Student_CGPA()
{
	return studentCGPA;
}

int StudentInfo::get_StudentYear()
{
	return studentYear;
}
```

```
void StudentInfo::print_StudentInfo() const
{
        cout << left;
   cout << "\t" << setw(32) << studentName ;
   cout << "  " << setw(21) << studentMatric ;
   cout << "  " << setw(10) << studentYear ;
   cout << "  " << setw(8) << fixed << setprecision(2) << studentCGPA << endl;
}

StudentInfo::~StudentInfo()
{
        studentName = "";
        studentMatric = "";
        studentCGPA = 0.00;
        studentYear = 0;
}

#endif
```

Linkedlist.h

```
#ifndef Included_Linkedlist_H
#define Included_Linkedlist_H

#include <iostream>
#include <string>
#include <iomanip>
#include <cassert>
#include <iterator>
#include "assert.h"
#include "StudentInfo.h"

using namespace std;

//*****************************************************************************************
**
// Create a struct using template which will hold all the student information using a class
//*****************************************************************************************
**
template <class Type>
struct nodeType
{
        Type info;
        nodeType<Type> *next;
};


//*****************************************************************************
// This class specifies the members to implement an iterator to a linked list.
//*****************************************************************************
template <class Type>
class linkedListIterator
{
  public:

    linkedListIterator();
        // Default constructor

    Type operator*();
        // To overload dereferencing operator*

    linkedListIterator<Type> operator++();
        // Overload the pre-increment operator

    bool operator==(const linkedListIterator<Type>& right) const;
        // Overload the equality operator

    bool operator!=(const linkedListIterator<Type>& right) const;
```

```
            // Overload the not equal to operator


  private:
     nodeType<Type> *current;
            // Pointer to point to the current node in the linked list
};


// Definition of the function of the class linkedListIterator
template <class Type>
linkedListIterator<Type>::linkedListIterator()
{
   current=NULL;
}


template <class Type>
Type linkedListIterator<Type>::operator* ()
{
   return current->info;
}


template <class Type>
linkedListIterator<Type> linkedListIterator<Type>::operator++()
{
   current=current->next;
   return *this;
}


template <class Type>
bool linkedListIterator<Type>::operator ==(const linkedListIterator<Type>& right) const
{
   return (current == right.current);
}


template <class Type>
bool linkedListIterator<Type>::operator !=(const linkedListIterator<Type>& right) const
{
   return (current != right.current);
}


//**********************************
// Base class - linkedListType
//**********************************
```

```
template <class Type>
class linkedListType
{
   public:

      const linkedListType<Type>& operator =(const linkedListType<Type>& );
                    // Overload the assignment operator

      void initializeList();
                    // Initialize the list to an empty state

                    bool isEmptyList() const;
                    // To determine whether the list is empty

      void print() const;
                    // To output the data contained in each node

      int length() const;
                    // To return the no of nodes in the list

      void destroyList();
                    // To delete all the nodes from the list

      Type front() const;
                    // To return the first element of the list

      Type back() const;
                    // To return the last element of the list

      linkedListIterator<Type> begin();
      // To return an iterator at the beginning of the linked list

      linkedListIterator<Type> end();
      // To return an iterator such that current is set to NULL

      linkedListType();
                    // Default constructor

      linkedListType(const linkedListIterator<Type>& otherList);
                    // Copy constructor

      ~linkedListType();
                    // Destructor- to delete all the nodes from the list

   protected:

      int num;
                    // Variable to store the no. of elements in the list
```

```
    nodeType<Type> *first;
                // Pointer to the first node of the list

    nodeType<Type> *last;
                // Pointer to the last node of the list

  private:

    void copyList(const linkedListType<Type>& otherList);
                // To make a copy of otherList

                nodeType<Type> *next;
};


template <class Type>
bool linkedListType<Type>::isEmptyList()const
{
  return (first=NULL);
}

template <class Type>
linkedListType<Type>::linkedListType()
{
  first=NULL;
  last=NULL;
  num=0;
}

template <class Type>
void linkedListType<Type>::destroyList()
{
  nodeType<Type> *temp; // Pointer to deallocate memory occupied by the node

  while(first!=NULL) // wWile there are nodes in the list
  {
    temp=first; // Set temp to current node
    first=first->next; // Advance first to the next node
    delete temp; // Deallocate the memory occupied by temp
  }

        last=NULL; // Initialize last to NULL, first has been set to NULL by while loop
  num=0;
}


template <class Type>
void linkedListType<Type>::initializeList()
{
```

```
    destroyList(); // Delete if the list has any nodes
}

template <class Type>
void linkedListType<Type>::print() const
{
    cout << "\tStudent Informations " << endl;
    cout << "..............................................................................." << endl;

    nodeType<Type> *current=first;

            cout << "\tStudent name\t\t\t" << "Matric number\t" << "\tYear" << "\t     CGPA" << endl;

    while(current!=NULL)
            {
                cout << left;
        cout << "\t" << setw(32) << current-> info.get_StudentName() ;
        cout << "  " << setw(21) << current-> info.get_StudentMatric() ;
        cout << "  " << setw(10) << current-> info.get_StudentYear() ;
        cout << "  " << setw(8) << fixed << setprecision(2) << current-> info.get_Student_CGPA() << endl;
        current=current->next;
    }// end while
        cout << "..............................................................................." << endl;

}


template <class Type>
int linkedListType<Type>::length()const
{
    return num;
}


template <class Type>
Type linkedListType<Type>::front() const
{
    assert(first!= NULL);
    return first;
}


template <class Type>
Type linkedListType<Type>::back() const
{
    assert(last!=NULL);
    return last;
}
```

```
template <class Type>
linkedListIterator<Type> linkedListType<Type>::begin()
{
   linkedListIterator<Type> temp(first);
   return temp;
}


template <class Type>
linkedListIterator<Type> linkedListType<Type>::end()
{
   linkedListIterator<Type> temp(NULL);
   return temp;
}


template <class Type>
void linkedListType<Type>::copyList(const linkedListType<Type>& otherList)
{
   nodeType<Type> *newNode; // pointer to create a node
   nodeType<Type> *current; // pointer to traverse the list

   if(first!=NULL) // make the list empty if it is nonempty
      destroyList();

            if(otherList.first == NULL) //otherList is empty
   {
      first=NULL;
      last=NULL;
      num=0;
   }

   else
   {
      current=otherList.first; // current points to the list to be copied
      num = otherList.count;

      first = new nodeType <Type>; //copy the first node and create the node
      first->info= current->info;
      first->next=NULL;
      last=first;
      current=current->next;

   while (current!=NULL) // copy the remaining list
   {
      newNode = new nodeType<Type>;
      newNode->info = current->info;
      newNode->next = NULL;
```

```
        last->next = newNode;
        last = newNode;

        current=current->link;
     }//end while

   }//end else

}//end copylist

template <class Type>
linkedListType<Type>::~linkedListType()
{
    destroyList();
}

template <class Type>
linkedListType<Type>::linkedListType(const linkedListIterator<Type>& otherList)
{
     first = NULL;
     copyList(otherList);
}

template <class Type>
const linkedListType<Type>& linkedListType<Type>::operator= (const linkedListType<Type>& otherList)
{
      if (this != &otherList) // to avoid self-copy
      {
         copyList(otherList);
      }
      return *this;
}

//*******************************************
// Derived class - Unordered Linked List
//*******************************************
template <class Type>
class unorderedLinkedList: public linkedListType<Type>
{
    public:
        bool search(const string searchItem) const;
                        // To determine whether searchItem is in the list

        void insertFirst(const Type& newItem);
                        // To insert newItem at the beginning of the list

        void insertLast(const Type& newItem);
                        //To insert newItem at the end of the list
```

```
    void deleteNode(const string deleteItem);
                    // To delete deleteItem from the list

  private:

          int num;
          // Variable to store the no. of elements in the list
};


template <class Type>
bool unorderedLinkedList<Type>::search(const string searchItem) const
{
  bool found = false;

  nodeType<Type> *current; //pointer to traverse the list

  current =this->first;  //start the search at the first node

  while (current != NULL && !found)
          {
    if (current-> info.get_StudentMatric() == searchItem)
      found = true;
    else
      current = current->next;
        }// end while

          if (found)
          {
                    cout << "\tStudent information found in Unordered Linked List" << endl;
    cout << "..................................................................................." << endl;
                    cout << "\tStudent name\t\t\t" << "Matric number\t" << "\tYear" << "\t     CGPA" << endl;

    cout << left;
    cout << "\t" << setw(32) << current-> info.get_StudentName() ;
    cout << "  " << setw(21) << current-> info.get_StudentMatric() ;
    cout << "  " << setw(10) << current-> info.get_StudentYear() ;
    cout << "  " << setw(8) << fixed << setprecision(2) << current-> info.get_Student_CGPA() << endl;
    cout << "..................................................................................." << endl;
          }// end if

          else
                    cout << "\tNo data found in the unordered linked list...";


  return found;
}//end search
```

```
template <class Type>
void unorderedLinkedList<Type>::insertFirst(const Type& newItem)
{
   nodeType<Type> *newNode; // pointer to create the new node
   newNode = new nodeType<Type>; // create the new node
   nodeType<Type> *first;
   nodeType<Type> *last;
   newNode->info = newItem;

   newNode->next=this->first; //insert newnode
   this->first=newNode;

   num++;

   if(this->last == NULL) // if list is empty, newNode is the last node in the list
         this->last=newNode;

}// end insertFirst


template <class Type>
void unorderedLinkedList<Type>::insertLast(const Type& newItem)
{
   nodeType<Type> *newNode;
         newNode = new nodeType<Type>;
         newNode->info = newItem; //store the new item in the node
         newNode->next = NULL; //set the link field of newNode to NULL
         nodeType<Type> *first;
         nodeType<Type> *last;

   if (first == NULL)
   {
      first= newNode;
      last= newNode;
      num++;
   }// end if

   else
   {
      last->next= newNode;
      last= newNode;
      num++;
   }
}// end insertLast


template <class Type>
void unorderedLinkedList<Type>::deleteNode(const string deleteItem)
```

```
{
   nodeType<Type> *current; // pointer to traverse the list
   nodeType<Type> *trailCurrent; // pointer just before current
   nodeType<Type> *first;
   nodeType<Type> *last;
   bool found;

   if (this->first == NULL) //Case 1; the list is empty.
      cout << "\n\tUnable to delete from an empty unordered linked list." << endl;

   else
   {
      //Case 2  The node need to be deleted is the 1st node
      if (this->first-> info.get_StudentMatric() == deleteItem)
       {
         current = this->first;
         this->first = this->first->next;
         num--;
         if (this->first == NULL) // list has only one node
            this->last = NULL;
         delete current;
       }

       else // search the list for the node with given info
       {
          found = false;
          trailCurrent= this->first;//set trailCurrent points to 1st node
          current= this->first->next;//set current points to 2nd node

          while (current !=NULL && !found)
          {
             if (current->info.get_StudentMatric() != deleteItem)
             {
                trailCurrent= current;
                current= current->next;
             }

             else
               found = true;
          }// end while

          if(found)//if found, delete the node
          {
             trailCurrent->next = current->next;
             num--;

             if (this->last == current)//Case 3, node to be deleted was the last node
               this->last = trailCurrent; //update the value of last
```

```
            delete current;//delete the node from the list

            cout << "\n\tThe student info with matric number " <<  deleteItem << " deleted from the Unordered
Linked List." << endl;
        }

        else
          cout << "\n\tThe student info with matric number " <<  deleteItem << " is not in the Unordered Linked
List." << endl; // case4
      }//end else
   }//end else
}//end deleteNode



//*****************************************
// Derived class - Ordered Linked List
//*****************************************
template <class Type>
class orderedLinkedList: public linkedListType<Type>
{
        public:

                bool search(const string searchItem) const;
                //Function to determine whether searchItem is in the list.

                void insert(const Type& newItem);
                //Function to insert newItem in the list.

                void insertFirst(const Type& newItem);
                //Function to insert newItem at the beginning of the list.

                void insertLast(const Type& newItem);
                //Function to insert newItem at the end of the list.

                void deleteNode(const string deleteItem);
                //Function to delete deleteItem from the list.

        private:

        int num;
        // Variable to store the no. of elements in the list
};


template <class Type>
bool orderedLinkedList<Type>::search(const string searchItem) const
{
   bool found = false;
```

```
   nodeType<Type> *current; //pointer to traverse the list

   current =this->first;  //start the search at the first node

   while (current != NULL && !found)
           {
      if (current-> info.get_StudentMatric() == searchItem)
         found = true;

      else
         current = current->next;
           }

           if (found)
           {
                        cout << "\tStudent information found in Ordered Linked List" << endl;
      cout << "......................................................................................" << endl;
                        cout << "\tStudent name\t\t\t" << "Matric number\t" << "\tYear" << "\t     CGPA" << endl;

      cout << left;
      cout << "\t" << setw(32) << current-> info.get_StudentName() ;
      cout << "  " << setw(21) << current-> info.get_StudentMatric() ;
      cout << "  " << setw(10) << current-> info.get_StudentYear() ;
      cout << "  " << setw(8) << fixed << setprecision(2) << current-> info.get_Student_CGPA() << endl;
      cout << "......................................................................................" << endl;

           }

           else
                        cout << "\tNo data found in the ordered linked list...";


   return found;
}//end search


template <class Type>
void orderedLinkedList<Type>::insert(const Type& newItem)
{
   nodeType<Type> *current; //pointer to traverse the list
   nodeType<Type> *trailCurrent; //pointer just before current
   nodeType<Type> *newNode;  //pointer to create a node

   bool  found;

   newNode = new nodeType<Type>; //create the node
   newNode->info = newItem;   //store newItem in the node
   newNode->next = NULL;      //set the link field of the node to NULL
```

```
    if (this->first == NULL)  //Case 1
    {
        this->first = newNode;
        this->last = newNode;
        this->num++;
    }

    else
    {
        current = this->first;
        found = false;

        while (current != NULL && !found) //search the list

          if (current-> info.get_StudentMatric() >= newNode -> info.get_StudentMatric())
              found = true;
          else
          {
              trailCurrent = current;
              current = current->next;
          }

        if (current == this->first)      //Case 2
        {
            newNode->next = this->first;
            this->first = newNode;
            this->num++;
        }

        else                     //Case 3
        {
            trailCurrent->next = newNode;
            newNode->next = current;

            if (current == NULL)
                this->last = newNode;

            this->num++;
        }

    }//end else

}//end insert


template<class Type>
void orderedLinkedList<Type>::insertFirst(const Type& newItem)
{
```

```
   insert(newItem);
}//end insertFirst


template<class Type>
void orderedLinkedList<Type>::insertLast(const Type& newItem)
{
   insert(newItem);
}//end insertLast


template<class Type>
void orderedLinkedList<Type>::deleteNode(const string deleteItem)
{
   nodeType<Type> *current; // pointer to traverse the list
   nodeType<Type> *trailCurrent; // pointer just before current
   nodeType<Type> *first;
   nodeType<Type> *last;
   bool found;

   if (this->first == NULL) //Case 1; the list is empty.
      cout << "\n\tUnable to delete from an empty ordered linked list." << endl;

   else
   {
      //Case 2  The node need to be deleted is the 1st node
      if (this->first-> info.get_StudentMatric() == deleteItem)
       {
         current = this->first;
         this->first = this->first->next;
         num--;
         if (this->first == NULL) // list has only one node
            this->last = NULL;
         delete current;
       }

       else // search the list for the node with given info
       {
          found = false;
          trailCurrent= this->first;//set trailCurrent points to 1st node
          current= this->first->next;//set current points to 2nd node

          while (current !=NULL && !found)
          {
             if (current->info.get_StudentMatric() != deleteItem)
             {
                trailCurrent= current;
                current= current->next;
             }
```

```
            else
              found = true;
          }// end while

          if(found)//if found, delete the node
          {
            trailCurrent->next = current->next;
            num--;

            if (this->last == current)//Case 3, node to be deleted was the last node
              this->last = trailCurrent; //update the value of last

            delete current;//delete the node from the list

            cout << "\n\tThe student info with matric number " <<  deleteItem << " deleted from the Ordered Linked
List." << endl;
          }
          else
            cout << "\n\tThe student info with matric number " <<  deleteItem << " is not in the Ordered Linked List."
<< endl; // case4

                    }//end else

  }//end else

}// end deleteNode
#endif
```

Test Case

Main Menu:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        This program is to store and organize students' information
==================================================================================

        Student's Information:

         # Add ------------------------------------------------------ Enter '1'

         # Delete ---------------------------------------------------- Enter '2'

         # Search & Display ------------------------------------------ Enter '3'

         # Display Full Information List ----------------------------- Enter '4'

         # Exit Program ---------------------------------------------- Enter '5'

==================================================================================

        Choice:
```

User key in other than 1~5 in main menu:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Invalid input...

        Please enter 1 to 5 ONLY

        Press "Enter" to return to Menu...

Press any key to continue . . .
```

User key in ‘1’ in main menu:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Add Student Information:
==========================================================================================

        -> Name:
```

User key in all information with correct format:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Add Student Information:
==========================================================================================

        -> Name: Dolly
        -> Matric Number: l148y48
        -> CGPA: 3.98
        -> Year of study: 1
Press any key to continue . . .
```

User key in information with an invalid CGPA (CGPA less than zero, more than four or not a numerical value):

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Add Student Information:
==================================================================================

        -> Name: Nemo
        -> Matric Number: 774nm
        -> CGPA: xxx
           Invalid CGPA...
           Re-enter CGPA: -0.5
           Invalid CGPA...
           Re-enter CGPA: 6.2
           Invalid CGPA...
           Re-enter CGPA: 2.65
        -> Year of study: 3
Press any key to continue . . .
```

User key in information with an invalid year of study(year less than one, more than five or not a numerical value):

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Add Student Information:
==================================================================================

        -> Name: Bob
        -> Matric Number: bob4524
        -> CGPA: 0.97
        -> Year of study: 0
           Invalid Year of Study...
           Re-enter Year of Study: 9
           Invalid Year of Study...
           Re-enter Year of Study: s
           Invalid Year of Study...
           Re-enter Year of Study: 5
Press any key to continue . . .
```

User key in ‘2’ in main menu:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System



        Student Informations
...................................................................................
        Student name                    Matric number           Year        CGPA
        Popeye                            1p2p3y                 2          1.23
        Nemo                              774nm                  3          2.65
        Bob                               bob4524                5          0.97
        Dolly                             ll48y48                1          3.98
        Scooby                            scb6666                4          3.88
...................................................................................



        Delete Student Information:
===================================================================================
        Matric number: _
```

User key in a matric number when the list is empty:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System



        Student Informations
...................................................................................
        Student name                    Matric number           Year        CGPA
...................................................................................



        Delete Student Information:
===================================================================================
        Matric number: hey001

        Unable to delete from an empty unordered linked list.

        Unable to delete from an empty ordered linked list.
Press any key to continue . . . _
```

User key in an invalid matric number:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System



        Student Informations
...................................................................................
        Student name                    Matric number           Year        CGPA
        Popeye                            1p2p3y                 2          1.23
        Nemo                              774nm                  3          2.65
        Bob                               bob4524                5          0.97
        Dolly                             ll48y48                1          3.98
        Scooby                            scb6666                4          3.88
...................................................................................



        Delete Student Information:
===================================================================================
        Matric number: 33aspy

        The student info with matric number 33aspy is not in the Unordered Linked List.

        The student info with matric number 33aspy is not in the Ordered Linked List.
Press any key to continue . . .
```

User key in a valid matric number:

```
C:\Users\Dell\Desktop\main.exe



                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System



        Student Informations
...................................................................................
        Student name                    Matric number           Year        CGPA
        Popeye                            1p2p3y                 2          1.23
        Nemo                              774nm                  3          2.65
        Bob                               bob4524                5          0.97
        Dolly                             ll48y48                1          3.98
        Scooby                            scb6666                4          3.88
...................................................................................



        Delete Student Information:
===================================================================================
        Matric number: 774nm

        The student info with matric number 774nm deleted from the Unordered Linked List.

        The student info with matric number 774nm deleted from the Ordered Linked List.
Press any key to continue . . .
```

User key in ‘3’ in main menu:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Search Student Information:
================================================================================
        Matric number:
```

User key in an invalid matric number:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Search Student Information:
================================================================================
        Matric number: hebat666


        No data found in the unordered linked list...

        No data found in the ordered linked list...

Press any key to continue . . .
```

User key in a existing matric number:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Search Student Information:
==========================================================================================
        Matric number: scb6666


        Student information found in Unordered Linked List
..........................................................................................
        Student name                    Matric number           Year        CGPA
        Scooby                            scb6666                4          3.88
..........................................................................................


        Student information found in Ordered Linked List
..........................................................................................
        Student name                    Matric number           Year        CGPA
        Scooby                            scb6666                4          3.88
..........................................................................................


Press any key to continue . . .
```

User key in ‘4’ in main menu:

```
C:\Users\Dell\Desktop\main.exe

                    WELCOME TO UNIVERSITI SAINS MALAYSIA
                        Student Information System


        Display Student Information:
==========================================================================================

        Print Student Information using Ordered Linked List
        Student Informations
..........................................................................................
        Student name                    Matric number           Year        CGPA
        Popeye                            1p2p3y                 2          1.23
        Bob                               bob4524                5          0.97
        Dolly                             1l48y48                1          3.98
        Scooby                            scb6666                4          3.88
..........................................................................................


        Print Student Information using Unordered Linked List
        Student Informations
..........................................................................................
        Student name                    Matric number           Year        CGPA
        Popeye                            1p2p3y                 2          1.23
        Scooby                            scb6666                4          3.88
        Bob                               bob4524                5          0.97
        Dolly                             1l48y48                1          3.98
..........................................................................................
Press any key to continue . . .
```

User key in ‘5’ in the main menu: