```
//Abdellah Amrhar
/* Program that uses linked lists to
Add/Delete/Search/Display nodes */


#include <iostream>
using namespace std;


//node structure to hold student info
struct node
{
   int ID;                  //variable to hold student IDs
   string name;               //variable to hold student name
   int age;                  //variable to hold student age
   node *nxt;                  //Point to next node
};
   node *start_ptr = NULL;     //Initialize start pointer to null

//Function prototypes
void menu(int&);
void add_node();
void delete_node();
void search_node();
void display();

int main()
{
   //Character to repeat program
   char repeat;

   do{
         //Menu option
         int choice;

         cout << "\n\nLinked List Menu\n"
             << "================\n"
             << "1. Add a Node\n"
             << "2. Delete a Node\n"
             << "3. Search for a Node\n"
             << "4. Display entire Linked List\n";

         // Respond to user's menu selection
         menu(choice);

         switch(choice)
         {
```

```
            case 1:
                {
                  //Add a node anywhere in LL
                  add_node();
                  display();
                }
                  break;
            case 2:
                  //Delete a node anywhere in LL
                 {
                  delete_node();
                  display();
                 }
                  break;
            case 3:
                  //Search for a node in LL
                 {
                  search_node();
                   display();
                 }
                  break;

            case 4:
                  //Display entire LL
                  display();
                  break;
          }



       cout << "\nDo you want to repeat the menu? <Y/N>: ";
       cin  >> repeat;

       }while(repeat == 'y' || repeat == 'Y');

    return 0;
}
//Menu function
void menu(int &a)
{

    cout << "\nEnter a menu choice: ";
    cin  >> a;
       while (a < 1 || a > 4)
       {
          cout << "\nInvalid entry. Re-enter menu choice: ";
          cin  >> a;
       }
```

```
}
//Function to add a node anywhere in LL
void add_node()
{
   int num;                    //User-inputted ID
   node *newNode;                 // A new node
   node *nodePtr;                // To traverse the list
   node *previousNode = NULL;      // The previous node

   //Allocate a new node and store num there
   newNode = new node;

   cout << "\nEnter student ID: ";
   cin  >> num;
   cout << "\nEnter student Name: ";
   cin  >> newNode->name;
   cout << "\nEnter student Age: ";
   cin  >> newNode->age;


   newNode -> ID = num;  //Allocate a new node and store num there

   //If there are no nodes in the list make newNode the first node
   if (!start_ptr)
   {
      start_ptr = newNode;
      newNode -> nxt = NULL;
   }
   else    //otherwise, insert newNode
   {
      //Position nodePtr at the head of list
      nodePtr = start_ptr;

      //Initialize previousNode to NULL
      previousNode = NULL;

      //Skip all nodes whose ID is less than num
      while (nodePtr != NULL && nodePtr -> ID < num)
      {
         previousNode = nodePtr;
         nodePtr = nodePtr -> nxt;
      }

      //If the new node is to be the 1st in the list,
      //insert it before all other nodes
      if (previousNode == NULL)
      {
         start_ptr = newNode;
```

```
            newNode -> nxt = nodePtr;
        }
        else    //otherwise insert after the previous node
        {
            previousNode -> nxt = newNode;
            newNode -> nxt = nodePtr;
        }
    }

}
//Function to delete node anywhere
void delete_node()
{
    int num;            //User-inputted ID
    node *nodePtr;      //To traverse the list
    node *previousNode; //To point to the previous node

    cout << "\nWhich student ID do you want to delete?: ";
    cin  >> num;

    //If LL is empty, do nothing
    if (!start_ptr)
    {
        cout << "\n The list is empty.\n";
        return;
    }
    //Determine if the first node is the one
    if (start_ptr -> ID == num)
    {
        nodePtr = start_ptr -> nxt;
        delete start_ptr;
        start_ptr = nodePtr;
        cout << "\nStudent ID " << num << " has been deleted!" << endl;
    }
    else
    {
        //Initialize nodePtr to start_ptr of list
        nodePtr = start_ptr;

        //Skip all nodes whose ID member is not equal to num
        while (nodePtr != NULL && nodePtr -> ID != num)
        {
            previousNode = nodePtr;
            nodePtr = nodePtr -> nxt;
        }
        //If nodePtr is not at the end of list,
        //link the previous node to the node after
        //nodePtr, then delete nodePtr
```

```
        if (nodePtr != NULL)
        {
            previousNode -> nxt = nodePtr -> nxt;
            delete nodePtr;
            cout << "\nStudent ID " << num << " has been deleted!" << endl;
        }
        //If nodePtr is at end of list it means that
        //student ID was not found
        else if (nodePtr == NULL)
        {
            cout << "\nStudent ID " << num << " not found." << endl;
        }
    }
}
//Function to search node anywhere in the LL
void search_node()
{
    int num;            //User-inputted student ID
    node *nodePtr;      //To move through the list

    //Position NodePtr at start_ptr
    nodePtr = start_ptr;

    cout << "\nWhat student ID do you want to look for?: ";
    cin  >> num;

    //While nodePtr points to a node, traverse the list
    //until you find the target
    while (nodePtr)
    {
        if (nodePtr -> ID == num)
        {
            cout << "\nStudent ID " << num << " was found." << endl;
            cout << "\nID:  " << nodePtr -> ID <<endl;
            cout << "\nName:  " << nodePtr -> name<<endl;
            cout << "\nAge:  " << nodePtr -> age<<endl;


            return;
        }
        else
            nodePtr = nodePtr -> nxt;
    }
    if (nodePtr == NULL)
        cout << "\nStudent ID " << num << " was not found." << endl;
}
//Function to display entire contents of LL
void display()
```

```
{
   node *temp1, *temp2;     //Temporary pointers
   temp1 = start_ptr;

   cout << "\nLinked List Nodes:\n"
        << "====================\n";
   do{
      if (temp1 == NULL)
         cout << "\nThe list is empty." << endl;
      else
      {
         //Display details for what temp points to
         cout << "\nID:  " << temp1 -> ID<<endl ;
         cout << "\nName:  " << temp1 -> name<<endl;
         cout << "\nAge:  " << temp1 -> age<<endl;
         cout<<"-------------"<< endl;

         //Move to next node (if present)
         temp1 = temp1 -> nxt;
      }
    }while(temp1 != NULL);


}
```