COMSATS University Islamabad, Vehari Campus

Name:	**Uzair Mukhtar**
Registration Number:	**SP22-BCS-081**
Department:	**Computer Science**
Course:	**DSA**
Instructor:	**Dr Salman Iqbal**
Assignment:	**1st**
Section:	**B**

Activity 1

function to display linked list

Solution:

C++

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

class Node
{
      public:
            int data;
            Node* next;
            Node(int data)
            {
                  this->data=data;
                  this->next=NULL;
            }
};
//Print using while loop
void print(Node* &head)
{
Node* temp=head;
   cout<<"  Linked List is:\n  ";
      while(temp!=NULL)
      {
            cout<<temp->data<<" ";
            temp=temp->next;
      }
      cout<<endl;

Node* ptr=head;
   //cout<<"Linked List is:\n";
   cout<<"   ****head address:**** "<<&head<<endl;
   cout<<"-----------------------------------------------------\n";
   cout<<"   head content:"<<head<<endl;
   cout<<"-----------------------------------------------------\n";
   cout<<"   ****ptr address:**** "<<&ptr<<endl;
   cout<<"-----------------------------------------------------\n";
   cout<<"   ptr content:"<<ptr<<endl;
   cout<<"-----------------------------------------------------\n";
   cout<<"   ptr->data: "<<ptr->data<<endl;
   cout<<"-----------------------------------------------------\n";
   ptr=ptr->next;
      while(ptr!=NULL)
      {
```

```
            cout<<"   ptr: "<<ptr<<endl;
            cout<<"   ptr->next: "<<ptr->next<<endl;
            cout<<"   ptr->data: "<<ptr->data<<endl;

cout<<"-----------------------------------------------------\n";
            ptr=ptr->next;
      }
}
int main()
{
      Node* head=NULL;
      Node* node1=new Node(1);
      head=node1;
      Node* node2=new Node(2);
      node1->next=node2;
      Node* node3=new Node(20);
      node2->next=node3;
      Node* node4=new Node(30);
      node3->next=node4;

      print(head);
}
```

```
G:\My Drive\DSA\mindless.exe
 Linked List is:
 1 2 20 30
  ****head address:**** 0x6ffde8
-----------------------------------------------------
  head content:0x1e1430
-----------------------------------------------------
  ****ptr address:**** 0x6ffd90
-----------------------------------------------------
  ptr content:0x1e1430
-----------------------------------------------------
  ptr->data: 1
-----------------------------------------------------
  ptr: 0x1e1480
  ptr->next: 0x1e1810
  ptr->data: 2
-----------------------------------------------------
  ptr: 0x1e1810
  ptr->next: 0x1e1830
  ptr->data: 20
-----------------------------------------------------
  ptr: 0x1e1830
  ptr->next: 0
  ptr->data: 30
-----------------------------------------------------
```

Activity 2(a)

Menu Singly Linked List:

Solution:

C++

```
#include <iostream>
#include <conio.h>
using namespace std;


class Node {
      public:
    int data;
    Node* next;

    Node(int data)
    {
      this->next=NULL;
      this->data=data;

      }
};
void seek(Node* head) {
    int target;
    std::cout << "Enter the value you want to seek: ";
    std::cin >> target;

    Node* current = head;
    int flag=0;

    while (current != NULL) {
        if (current->data == target) {
            std::cout << "Value " << target << " found in the list." <<
std::endl;
            flag++;
        }
        current = current->next;
    }

    if(flag==0)
    {
      std::cout << "Value " << target << " not found in the list." <<
std::endl;
      }

}
```

```
void reverseAndPrint(Node*& head) {
    Node* prev = NULL;
    Node* current = head;
    Node* next = NULL;

    while (current != NULL) {
        next = current->next;  // Store the next node.
        current->next = prev;  // Reverse the current node's pointer to
the previous node.
        prev = current;        // Move the 'prev' pointer to the
current node.
        current = next;        // Move the 'current' pointer to the
next node.
    }

    head = prev;  // Update the head to point to the new first node
(previous last node).

    // Print the reversed list.
    cout<<"reversed linked list is:";
    current = head;
    while (current != NULL) {
        std::cout << current->data << " -> ";
        current = current->next;
    }
    std::cout << "NULL" << std::endl;
}

void deleteAtEnd(Node*& head) {
    if (head == NULL) {
        cout << "List is empty. Cannot delete from an empty list." <<
endl;
        return;
    }

    if (head->next == NULL) {
        // If there is only one node in the list, delete it and set
head to nullptr.
        delete head;
        head = NULL;
        cout<<"deleted successfully!\n";
        return;
    }

    Node* current = head;
    while (current->next->next != NULL) {
        current = current->next;
    }
```

```
    delete current->next;
    current->next = NULL;
    cout<<"deleted successfully!\n";
}

void deleteAtBegining(Node*& head) {
    if (head == NULL) {
        cout << "List is empty. Cannot delete from an empty list." <<
std::endl;
        return;
    }

    Node* temp = head;
    head = head->next;
    delete temp;
    cout<<"deleted successfully!\n";
}

void deleteAnyValue(Node* &head)
{
if(head==NULL)
     {
      cout<<"Linked List is empty so Deletion not Possible";
      return;
       }
int V,data;
     cout<<"For deletion At Value, Enter Any Value in Link List:";
     cin>>V;

     int flag=0;
      Node* temp=head;
      while(temp->next->data!=V  && temp->next->next!=NULL)
      {
            temp=temp->next;
      }
      if(temp->next->data==V)
      {
            Node* ptr=temp->next;
            temp->next=temp->next->next;
            delete ptr;
            flag++;
      }
            if(head->data==V)
      {
            Node* ptr=head;
            head=head->next;
            delete ptr;
            flag++;
      }
      if(flag==0)
```

```
      {
            cout<<"Value does not exist in the Linked list\n";
      }
}


void insertAtAnyValue(Node* &head)
{
     int V,data;
     cout<<"For insertion At Value, Enter Any Value in Link List:";
     cin>>V;
     int flag=0;
      Node* temp=head;
      while(temp->next->data!=V  && temp->next->next!=NULL)
      {
            temp=temp->next;
      }
      if(temp->next->data==V)
      {   cout<<"Enter value of Node:";
        cin>>data;
            Node* ptr=new Node(data);
            ptr->next=temp->next;
            temp->next=ptr;
            cout<<"inserted successfully!\n";
            flag++;
      }
            if(head->data==V)
      {   cout<<"Enter value of Node:";
        cin>>data;
            Node* ptr=new Node(data);
            ptr->next=head;
            head=ptr;
            cout<<"inserted successfully!\n";
            flag++;
      }
      if(flag==0)
      {
            cout<<"Value does not exist in the Linked list and not
inserted successfully!\n";
      }
}

void insertAtEnd(Node* &head,int data)
{
      if(head==NULL)
      {
            Node* node1=new Node(data);
      node1->next=head;
      head=node1;
      }
      else
```

```
      {
            Node* temp=head;
      while(temp->next!=NULL)
      {
            temp=temp->next;
      }
      Node* lastNode=new Node(data);
      temp->next=lastNode;

      }

}
void print(Node* &head)
{

     if(head==NULL)
    {
       cout<<"Linked List is Empty";
      }
      else
      {  cout<<"The items present in the List are:";
            Node* temp=head;
      while(temp!=NULL)
      {
            cout<<temp->data<<" ";
            temp=temp->next;
      }
      cout<<endl;
      }
}
void insertAtBeginingsingly(Node* &head,int data)
{
      Node* node1=new Node(data);
      node1->next=head;
      head=node1;
}

int main() {

    int operation;

    Node* singlyList = NULL; // head of singly linked list

        do {
            cout << "\n\nWhich operation you want to perform:" << endl;
            cout << "1: Insertion" << endl;
            cout << "2: Deletion" << endl;
            cout << "3: Display" << endl;
            cout << "4: Reverse" << endl;
            cout << "5: Seek" << endl;
            cout << "6: Back to Previous Menu" << endl;
```

```
            cout << "Enter your choice: ";
            cin >> operation;

            if (operation == 6) {
                break;  // exit
            }

            // Handle Insertion
            if (operation == 1) {
                int insertionOption;

                do {
                    cout << "\n\nInsertion Options:" << endl;
                    cout << "1: Insert at the beginning" << endl;
                    cout << "2: Insert at the end" << endl;
                    cout << "3: Insert at a specific data node" <<
endl;
                    cout << "4: Back to Previous Menu" << endl;
                    cout << "Enter your choice: ";
                    cin >> insertionOption;

                    if (insertionOption == 4) {
                        break;  // Return to the previous menu
                    }

                       switch (insertionOption) {
                        case 1:
                            {   int data;
                            cout<<"Enter the Value to insert:";
                            cin>>data;

insertAtBeginingsingly(singlyList,data);
                                                cout<<"inserted
successfully!\n";
                                                print(singlyList);
                                                char c;
                                                cout<<"Press any key to
continue...";
                                                getch();
                             break;
                                            }
                        case 2:
                            {   int data;
                            cout<<"Enter the Value to insert:";
                            cin>>data;

insertAtEnd(singlyList,data);
                                                cout<<"inserted
successfully!\n";
                                                print(singlyList);
                                                char c;
```

```
                                                cout<<"Press any key to
continue...";
                                                getch();
                                break;
                                            }
                            case 3:
                                { insertAtAnyValue(singlyList);
                                                print(singlyList);
                                                char c;
                                                cout<<"\nPress any key to
continue...";
                                                getch();
                                break;
                                            }
                            default:
                                {
                                 cout << "Invalid option. Please try again."
<< endl;
                                break;
                                            }
                        }


                    } while (true);
                }
                     if (operation == 2) {
                    int deletionOption;

                    do {
                        cout << "\n\nDeletion Options:" << endl;
                        cout << "1: Delete at the beginning" << endl;
                        cout << "2: Delete at the end" << endl;
                        cout << "3: Delete at a specific data node" <<
endl;
                        cout << "4: Back to Previous Menu" << endl;
                        cout << "Enter your choice: ";
                        cin >> deletionOption;

                        if (deletionOption == 4) {
                            break;  // Return to the previous menu
                        }

                           switch (deletionOption) {
                            case 1:
                                {

deleteAtBegining(singlyList);
                                                print(singlyList);
                                                char c;
```

```
                                                cout<<"Press any key to
continue...";
                                                getch();
                                break;
                                            }
                            case 2:
                                {
                                                deleteAtEnd(singlyList);
                                                print(singlyList);
                                                char c;
                                                cout<<"Press any key to
continue...";
                                                getch();
                                break;
                                            }
                            case 3:
                                { deleteAnyValue(singlyList);
                                                print(singlyList);
                                                char c;
                                                cout<<"\nPress any key to
continue...";
                                                getch();
                                break;
                                            }
                            default:
                                {
                                cout << "Invalid option. Please try again."
<< endl;
                                break;
                                            }
                        }


                    } while (true);
                }
            if(operation==3)
            {
                print(singlyList);
                }
                if(operation==4)
                {
                    reverseAndPrint(singlyList);
                }
            if(operation==5)
                {
                    seek(singlyList);
                }
            } while (true);
```

```
    return 0;
}
```

```
G:\My Drive\DSA\always.exe


Which operation you want to perform:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Back to Previous Menu
Enter your choice: 2


Deletion Options:
1: Delete at the beginning
2: Delete at the end
3: Delete at a specific data node
4: Back to Previous Menu
Enter your choice: 3
Linked List is empty so Deletion not PossibleLinked List is Empty
Press any key to continue...
```

Activity 2(b)

Menu Doubly Linked List:

Solution:

C++

```
#include <iostream>
#include <conio.h>
using namespace std;


class NodeDoubly {
      public:
    int data;
    NodeDoubly* next;
    NodeDoubly* prev;

    NodeDoubly(int data)
    {
      this->next=NULL;
```

```
      this->data=data;
      this->prev=NULL;

      }
};
void seek(NodeDoubly* head) {
    int targetData;
    std::cout << "Enter the value you want to seek: ";
    std::cin >> targetData;

    NodeDoubly* current = head;
    bool found = false;

    while (current != NULL) {
        if (current->data == targetData) {
            found = true;
            std::cout << "Value " << targetData << " found in the
list." << std::endl;
            return;
        }
        current = current->next;
    }

    if (!found) {
        std::cout << "Value " << targetData << " not found in the
list." << std::endl;
    }
}


void reverseAndPrint(NodeDoubly* &head) {
    if (head == NULL) {
        std::cout << "List is empty. Nothing to reverse and print." <<
std::endl;
        return;
    }

    NodeDoubly* current = head;

    while (current->next != NULL) {
        current = current->next;
    }

    std::cout << "Reversed List:" << std::endl;

    while (current != NULL) {
        std::cout << (current->next ? "<-" : "") << current->data << "
-> ";
        current = current->prev;
    }
```

```
    std::cout << "NULL" << std::endl;
}

void deleteSpecificDataNode(NodeDoubly*& head) {
    int targetData;
    std::cout << "Enter the value you want to delete: ";
    std::cin >> targetData;

    if (head == NULL) {
        std::cout << "List is empty. Cannot delete from an empty list."
<< std::endl;
        return;
    }

    if (head->data == targetData) {
        NodeDoubly* temp = head;
        head = head->next;

        if (head != NULL) {
            head->prev = NULL;
        }

        delete temp;
        return;
    }

    NodeDoubly* current = head;

    while (current != NULL) {
        if (current->data == targetData) {
            current->prev->next = current->next;

            if (current->next != NULL) {
                current->next->prev = current->prev;
            }

            delete current;
            return;
        }

        current = current->next;
    }

    std::cout << "Value " << targetData << " not found in the list." <<
std::endl;
}


void deleteAtEnd(NodeDoubly*& head) {
    if (head == NULL) {
        std::cout << "List is empty. Cannot delete from an empty list."
```

```
<< std::endl;
        return;
    }

    if (head->next == NULL) {
        // If there's only one node in the list, delete it and set head
to NULL.
        delete head;
        head = NULL;
        return;
    }

    NodeDoubly* current = head;
    while (current->next->next != NULL) {
        current = current->next;
    }

    delete current->next;
    current->next = NULL;
}

void deleteAtHead(NodeDoubly*& head) {
    if (head == NULL) {
        std::cout << "List is empty. Cannot delete from an empty list."
<< std::endl;
        return;
    }

    NodeDoubly* temp = head;
    head = head->next;

    if (head != NULL) {
        head->prev = NULL;
    }

    delete temp;
}

void insertAtSpecificDataNode(NodeDoubly*& head, int dataToInsert) {
    int dataAfter;
    std::cout << "Enter the value after which you want to insert: ";
    std::cin >> dataAfter;

    NodeDoubly* new_node = new NodeDoubly(dataToInsert);
    NodeDoubly* current = head;

    while (current != NULL) {
        if (current->data == dataAfter) {
            new_node->prev = current;
            new_node->next = current->next;
```

```
            if (current->next != NULL) {
                current->next->prev = new_node;
            }

            current->next = new_node;
            return;
        }

        current = current->next;
    }
}


void insertAtEnd(NodeDoubly*& head, int data) {
    NodeDoubly* new_node = new NodeDoubly(data);
    NodeDoubly* current = head;

    if (head == NULL) {
        head = new_node;
    } else {
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = new_node;
        new_node->prev = current;
    }
}

void Display(NodeDoubly* head) {
    NodeDoubly* current = head;

    while (current != NULL) {
        std::cout << (current->prev ? "<-" : "") << current->data << "
-> ";
        current = current->next;
    }

    std::cout << "NULL" << std::endl;
}

void insertAtHead(NodeDoubly*& head, int data) {
    NodeDoubly* new_node = new NodeDoubly(data);
    if (head ==NULL) {
        head = new_node;
    } else {
        new_node->next = head;
        head->prev = new_node;
        head = new_node;
    }
}
```

```
int main() {

    int operation;
    NodeDoubly*             singlyList=NULL;
    NodeDoubly* dhead = NULL; // head of singly linked list

        do {
            cout << "\n\nWhich operation you want to perform:" << endl;
            cout << "1: Insertion" << endl;
            cout << "2: Deletion" << endl;
            cout << "3: Display" << endl;
            cout << "4: Reverse" << endl;
            cout << "5: Seek" << endl;
            cout << "6: Back to Previous Menu" << endl;
            cout << "Enter your choice: ";
            cin >> operation;

            if (operation == 6) {
                break;  // exit
            }

            // Handle Insertion
            if (operation == 1) {
                int insertionOption;

                do {
                    cout << "\n\nInsertion Options:" << endl;
                    cout << "1: Insert at the beginning" << endl;
                    cout << "2: Insert at the end" << endl;
                    cout << "3: Insert at a specific data node" <<
endl;
                    cout << "4: Back to Previous Menu" << endl;
                    cout << "Enter your choice: ";
                    cin >> insertionOption;

                    if (insertionOption == 4) {
                        break;  // Return to the previous menu
                    }

                       switch (insertionOption) {
                        case 1:
                            {   int data;
                            cout<<"Enter the Value to insert:";
                            cin>>data;
                                            insertAtHead(dhead,data);
                                            cout<<"inserted
successfully!\n";
                                            Display(dhead);
                                            char c;
                                            cout<<"Press any key to
continue...";
```

```
                                                getch();
                                break;
                                            }
                            case 2:
                                {   int data;
                                cout<<"Enter the Value to insert:";
                                cin>>data;
                                                insertAtEnd(dhead,data);
                                                cout<<"inserted
successfully!\n";
                                                Display(dhead);
                                                char c;
                                                cout<<"Press any key to
continue...";
                                                getch();
                                break;
                                            }
                            case 3:
                                { int data;
                                cout<<"Enter the Value to insert:";
                                cin>>data;
                                                
insertAtSpecificDataNode(dhead,data);
                                                Display(dhead);
                                                char c;
                                                cout<<"\nPress any key to
continue...";
                                                getch();
                                break;
                                            }
                            default:
                                {
                                 cout << "Invalid option. Please try again."
<< endl;
                                break;
                                            }
                        }
                                                
                                                
                                                

                    } while (true);
                }
                     if (operation == 2) {
                    int deletionOption;

                    do {
                        cout << "\n\nDeletion Options:" << endl;
                        cout << "1: Delete at the beginning" << endl;
                        cout << "2: Delete at the end" << endl;
                        cout << "3: Delete at a specific data node" <<
```

```
endl;
                    cout << "4: Back to Previous Menu" << endl;
                    cout << "Enter your choice: ";
                    cin >> deletionOption;

                    if (deletionOption == 4) {
                        break;  // Return to the previous menu
                    }

                       switch (deletionOption) {
                        case 1:
                            {
                                                deleteAtHead(dhead);
                                                Display(dhead);
                                                char c;
                                                cout<<"Press any key to
continue...";
                                                getch();
                            break;
                                            }
                        case 2:
                            {
                                                deleteAtEnd(dhead);
                                                Display(dhead);
                                                char c;
                                                cout<<"Press any key to
continue...";
                                                getch();
                            break;
                                            }
                        case 3:
                            { deleteSpecificDataNode(dhead);
                                                Display(dhead);
                                                char c;
                                                cout<<"\nPress any key to
continue...";
                                                getch();
                            break;
                                            }
                        default:
                            {
                            cout << "Invalid option. Please try again."
<< endl;
                            break;
                                            }
                    }


                } while (true);
```

```
            }
        if(operation==3)
        {
            Display(dhead);
            }
            if(operation==4)
            {
                reverseAndPrint(dhead);
            }
        if(operation==5)
            {
                seek(dhead);
             }
        } while (true);

    return 0;
}
```

```
Which operation you want to perform:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Back to Previous Menu
Enter your choice: 1


Insertion Options:
1: Insert at the beginning
2: Insert at the end
3: Insert at a specific data node
4: Back to Previous Menu
Enter your choice: 2
Enter the Value to insert:10
inserted successfully!
10 -> NULL
Press any key to continue...

Insertion Options:
1: Insert at the beginning
2: Insert at the end
3: Insert at a specific data node
4: Back to Previous Menu
Enter your choice: 2
Enter the Value to insert:20
inserted successfully!
10 -> <-20 -> NULL
Press any key to continue...
```