\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data Structure Lab

CEN-391

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Practical Exam

Code :-

#include <iostream>

using namespace std;

struct LinkedList

{

    int data;

    LinkedList \*next;

    LinkedList \*prev;

};

LinkedList \*Create\_NewNode()

{

    LinkedList \*newnode = (LinkedList \*)malloc(sizeof(LinkedList));

    cout << "Enter The Element : ";

    cin >> newnode->data;

    newnode->next = nullptr;

    newnode->prev = nullptr;

    return newnode;

}

void Display(LinkedList \*Head, int size)

{

    cout << "Display...\n";

    if (size == 0)

    {

        cout << "Linked List Is Empty!\n";

        return;

    }

    cout << "|Head|";

    while (Head)

    {

        cout << "--|" << Head->data << "|";

        Head = Head->next;

    }

    cout << "--|Tail|\n";

}

void Insert\_At\_End(LinkedList \*&Head, LinkedList \*&Tail, int &size)

{

    cout << "Insert At End Operation Is Selected... \n";

    LinkedList \*newnode = Create\_NewNode();

    if (size == 0)

    {

        size++;

        Head = newnode;

        Tail = newnode;

        Display(Head, size);

        return;

    }

    if (newnode == nullptr)

    {

        cout << "Memory Not Assigned!\n";

        return;

    }

    size++;

    Tail->next = newnode;

    newnode->prev = Tail;

    Tail = Tail->next;

    Display(Head, size);

}

void Delete\_At\_End(LinkedList \*&Head, LinkedList \*&Tail, int &size)

{

    cout << "Delete At End Operation Is Selected... \n";

    if (size == 0)

    {

        cout << "Linked List Underflow!\n";

        return;

    }

    size--;

    LinkedList \*todelete = Tail;

    Tail = Tail->prev;

    Tail->next = nullptr;

    cout << todelete->data << "\n";

    delete todelete;

    if (size == 0)

    {

        Head == nullptr;

        Tail == nullptr;

    }

    Display(Head, size);

}

void Minimum\_Element\_In\_Linked\_List(LinkedList \*Head, int size)

{

    cout << "Minimum Element In Linked List Operation Is Selected... \n";

    if (size == 0)

    {

        cout << "Empty List!\n";

        return;

    }

    int Min = 1e9;

    LinkedList \*curr = Head;

    while (curr != nullptr)

    {

        if (Min > curr->data)

            Min = curr->data;

        curr = curr->next;

    }

    cout << "Minimum Element : " << Min << "\n";

    Display(Head, size);

}

void Bars()

{

    cout << "---------------------------------------------------------------\n";

}

bool Options(LinkedList \*&Head, LinkedList \*&Tail, int &size)

{

    int opt;

    cin >> opt;

    Bars();

    switch (opt)

    {

    case 1:

        Insert\_At\_End(Head, Tail, size);

        break;

    case 2:

        Delete\_At\_End(Head, Tail, size);

        break;

    case 3:

        Minimum\_Element\_In\_Linked\_List(Head, size);

        break;

    case 4:

        Display(Head, size);

        break;

    case 5:

        return 0;

        break;

    default:

        cout << "Invalid Input!\nTry Again!\n\n";

    }

    Bars();

    return 1;

}

void Menu()

{

    cout << "\n\_\_\_\_\_Operations\_On\_Doubly\_Linked\_List\_\_\_\_\_ \n";

    cout << "1.Insert At End. \n";

    cout << "2.Delete At End. \n";

    cout << "3.Print Minimum Element Of Linked List. \n";

    cout << "4.Display. \n";

    cout << "5.Exit. \n";

    cout << "\nEnter Your Choice : ";

}

int main()

{

    system("cls");

    cout << "\_\_\_Vicky\_Gupta\_20BCS070\_\_\_\n";

    LinkedList \*Head = nullptr, \*Tail = nullptr;

    int size = 0;

    while (true)

    {

        Menu();

        if (!Options(Head, Tail, size))

            break;

    }

    cout << "Exiting...\n";

    Bars();

    return 0;

}

Output :-

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated