**Muhammad Abdullah**

19F-0916 | SE(3A)

DATA STRUCTURES

ASSIGNMENT # 1 COURse

**Question # 1:**

**PROGRAM:**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode;

};

class Fun

{

public:

Fun()

{

Head = NULL;

}

Node \*Head;

void Add\_Given\_Data();

void Search\_Node\_Data(int val);

void Display();

};

void Fun::Add\_Given\_Data()

{

Node \*temp = NULL;

int data[11] = { 12, 34, 56, 7, 5, 4, 3, 8, 9, 89, 12 };

for (int i = 0; i < 11; i++)

{

if (Head == NULL)

{

temp = new Node;

temp->data = data[i];

temp->NextNode = NULL;

Head = temp;

}

else

{

Node \*current = Head;

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = data[i];

current->NextNode = temp;

temp->NextNode = NULL;

}

}

}

void Fun::Search\_Node\_Data(int val)

{

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

Node \*current = Head;

int counter = 1;

while (current != NULL)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

current = current->NextNode;

counter++;

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

while (temp != NULL)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Data |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 2 to Search Data in Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 3 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Given Values in the Question Will Be Inserted Automatically !!" << endl;

List.Add\_Given\_Data();

cout << endl << "All given data is added in the list, check it from display function !" << endl;

cout << endl;

system("pause");

break;

}

case 2:

{

cout << "Enter value to search in Link List : ";

cin >> val;

List.Search\_Node\_Data(val);

cout << endl;

system("pause");

break;

}

case 3:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**A screen shot of a computer

Description automatically generated**

**Question # 2:**

**PROGRAM:**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode;

};

class Fun

{

public:

Fun()

{

Head = NULL;

}

Node \*Head;

void Add\_Given\_Data();

void Add\_Given\_Data\_Middle();

void Search\_Node\_Data(int val);

void Display();

};

void Fun::Add\_Given\_Data()

{

Node \*temp = NULL;

int data[11] = { 12, 34, 56, 7, 5, 4, 3, 8, 9, 89, 12 };

for (int i = 0; i < 11; i++)

{

if (Head == NULL)

{

temp = new Node;

temp->data = data[i];

temp->NextNode = NULL;

Head = temp;

}

else

{

Node \*current = Head;

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = data[i];

current->NextNode = temp;

temp->NextNode = NULL;

}

}

}

void Fun::Add\_Given\_Data\_Middle()

{

Node \*temp = NULL, \*current = Head;

int opt = 6, counter = 11;

int Data[3] = { 111,222,333 };

if (Head == NULL)

{

cout << endl << "Create List First !!" << endl;

}

else

{

for (int i = 2; i >= 0; i--)

{

if (opt > 1 && opt < counter)

{

current = Head;

for (int j = 2; j < counter - 1; j++)

{

if (j == opt)

{

temp = new Node;

temp->data = Data[i];

temp->NextNode = current->NextNode;

current->NextNode = temp;

break;

}

current = current->NextNode;

}

}

}

cout << "Given Values in the Question Will Be Inserted in The Middle Automatically !!" << endl;

cout << endl << "All given data is added in the list, check it from display function !" << endl;

}

}

void Fun::Search\_Node\_Data(int val)

{

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

Node \*current = Head;

int counter = 1;

while (current != NULL)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

current = current->NextNode;

counter++;

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

while (temp != NULL)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Data |" << endl;

cout << " | Press 2 to Add Data in Middle |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 3 to Search Data in Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 4 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Given Values in the Question Will Be Inserted Automatically !!" << endl;

List.Add\_Given\_Data();

cout << endl << "All given data is added in the list, check it from display function !" << endl;

cout << endl;

system("pause");

break;

}

case 2:

{

List.Add\_Given\_Data\_Middle();

cout << endl;

system("pause");

break;

}

case 3:

{

cout << "Enter value to search in Link List : ";

cin >> val;

List.Search\_Node\_Data(val);

cout << endl;

system("pause");

break;

}

case 4:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**Text

Description automatically generated**

**A screenshot of a computer screen

Description automatically generated**

**Question # 3:**

**PROGRAM:**

**NOTE: THIS CODE CAN PERFORM ANY ADDITION OR DELETION SO IT CAN REDO ACCORDING TO USER’S WILL!**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode;

};

class Fun

{

public:

Fun()

{

Head = NULL;

}

Node \*Head;

void Add\_Node\_First(int val);

void Add\_Node\_Last(int val);

void Add\_Node\_Random(int val);

void Delete\_Node\_First();

void Delete\_Node\_Last();

void Delete\_Node\_Random();

void Search\_Node\_Data(int val);

void Display();

};

void Fun::Add\_Node\_First(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = NULL;

Head = temp;

}

else

{

temp = new Node;

temp->data = val;

temp->NextNode = current;

Head = temp;

}

}

void Fun::Add\_Node\_Last(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->NextNode = NULL;

}

else

cout << endl << "!!! Create Link List First !!!" << endl;

}

void Fun::Add\_Node\_Random(int val)

{

Node \*temp = NULL, \*current = Head;

int opt = 0, counter = 0;

cout << endl << "Enter Postion to add a Node : ";

cin >> opt;

if (Head == NULL)

{

cout << endl << "Create List First !!" << endl;

}

else

{

Node \*temp1 = Head;

while (temp1 != NULL)

{

counter++;

temp1 = temp1->NextNode;

}

if (opt == 1)

{

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = NULL;

Head = temp;

}

else

{

temp = new Node;

temp->data = val;

current = Head;

Head = temp;

temp->NextNode = current;

}

}

else if (opt == counter)

{

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->NextNode = NULL;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter - 1; i++)

{

if (i == opt)

{

temp = new Node;

temp->data = val;

temp->NextNode = current->NextNode;

current->NextNode = temp;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Delete\_Node\_First()

{

Node \*temp = NULL;

if (Head != NULL)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

free(temp);

cout << endl << "First Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Last()

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << "Last Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Random()

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

int opt = 0, counter = 0;

cout << "Enter the Position of Node to Delete it : ";

cin >> opt;

while (current != NULL)

{

counter++;

current = current->NextNode;

}

if (opt == 1)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt == counter)

{

current = Head;

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter - 1; i++)

{

if (i == opt)

{

temp = new Node;

temp = current->NextNode;

current->NextNode = temp->NextNode;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Search\_Node\_Data(int val)

{

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

Node \*current = Head;

int counter = 1;

while (current != NULL)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

current = current->NextNode;

counter++;

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

while (temp != NULL)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Node on Start |" << endl;

cout << " | Press 2 to Add Node on Last |" << endl;

cout << " | Press 3 to Add Node Randomly |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 4 to Delete First Node |" << endl;

cout << " | Press 5 to Delete Last Node |" << "\t\t\tSINGLY LINK LIST" << endl;

cout << " | Press 6 to Delete Random Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 7 to Search Data in Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 9 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_First(val);

break;

}

case 2:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Last(val);

break;

}

case 3:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Random(val);

break;

}

case 4:

{

List.Delete\_Node\_First();

cout << endl;

system("pause");

break;

}

case 5:

{

List.Delete\_Node\_Last();

cout << endl;

system("pause");

break;

}

case 6:

{

List.Delete\_Node\_Random();

cout << endl;

system("pause");

break;

}

case 7:

{

cout << "Enter value to search in Link List : ";

cin >> val;

List.Search\_Node\_Data(val);

cout << endl;

system("pause");

break;

}

case 9:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question # 4:**

**PROGRAM:**

**NOTE: THIS CODE CAN PERFORM ANY ADDITION OR DELETION SO IT CAN UNDO ACCORDING TO USER’S WILL!**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode;

};

class Fun

{

public:

Fun()

{

Head = NULL;

}

Node \*Head;

void Add\_Node\_First(int val);

void Add\_Node\_Last(int val);

void Add\_Node\_Random(int val);

void Delete\_Node\_First();

void Delete\_Node\_Last();

void Delete\_Node\_Random();

void Search\_Node\_Data(int val);

void Display();

};

void Fun::Add\_Node\_First(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = NULL;

Head = temp;

}

else

{

temp = new Node;

temp->data = val;

temp->NextNode = current;

Head = temp;

}

}

void Fun::Add\_Node\_Last(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->NextNode = NULL;

}

else

cout << endl << "!!! Create Link List First !!!" << endl;

}

void Fun::Add\_Node\_Random(int val)

{

Node \*temp = NULL, \*current = Head;

int opt = 0, counter = 0;

cout << endl << "Enter Postion to add a Node : ";

cin >> opt;

if (Head == NULL)

{

cout << endl << "Create List First !!" << endl;

}

else

{

Node \*temp1 = Head;

while (temp1 != NULL)

{

counter++;

temp1 = temp1->NextNode;

}

if (opt == 1)

{

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = NULL;

Head = temp;

}

else

{

temp = new Node;

temp->data = val;

current = Head;

Head = temp;

temp->NextNode = current;

}

}

else if (opt == counter)

{

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->NextNode = NULL;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter - 1; i++)

{

if (i == opt)

{

temp = new Node;

temp->data = val;

temp->NextNode = current->NextNode;

current->NextNode = temp;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Delete\_Node\_First()

{

Node \*temp = NULL;

if (Head != NULL)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

free(temp);

cout << endl << "First Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Last()

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << "Last Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Random()

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

int opt = 0, counter = 0;

cout << "Enter the Position of Node to Delete it : ";

cin >> opt;

while (current != NULL)

{

counter++;

current = current->NextNode;

}

if (opt == 1)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt == counter)

{

current = Head;

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter - 1; i++)

{

if (i == opt)

{

temp = new Node;

temp = current->NextNode;

current->NextNode = temp->NextNode;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Search\_Node\_Data(int val)

{

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

Node \*current = Head;

int counter = 1;

while (current != NULL)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

current = current->NextNode;

counter++;

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

while (temp != NULL)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Node on Start |" << endl;

cout << " | Press 2 to Add Node on Last |" << endl;

cout << " | Press 3 to Add Node Randomly |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 4 to Delete First Node |" << endl;

cout << " | Press 5 to Delete Last Node |" << "\t\t\tSINGLY LINK LIST" << endl;

cout << " | Press 6 to Delete Random Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 7 to Search Data in Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 9 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_First(val);

break;

}

case 2:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Last(val);

break;

}

case 3:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Random(val);

break;

}

case 4:

{

List.Delete\_Node\_First();

cout << endl;

system("pause");

break;

}

case 5:

{

List.Delete\_Node\_Last();

cout << endl;

system("pause");

break;

}

case 6:

{

List.Delete\_Node\_Random();

cout << endl;

system("pause");

break;

}

case 7:

{

cout << "Enter value to search in Link List : ";

cin >> val;

List.Search\_Node\_Data(val);

cout << endl;

system("pause");

break;

}

case 9:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question # 5:**

**PROGRAM:**

**NOTE: THIS PROGRAM CAN PERFORM ANY ACTION WITH DOUBLY LINK LIST**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode, \*PreviousNode;

};

class Fun

{

public:

Fun()

{

Head = NULL;

}

Node \*Head;

void Add\_Node\_First(int val);

void Add\_Node\_Last(int val);

void Add\_Node\_Random(int val);

void Delete\_Node\_First();

void Delete\_Node\_Last();

void Delete\_Node\_Random();

void Search\_Node\_Data(int val);

void Display();

};

void Fun::Add\_Node\_First(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = NULL;

temp->PreviousNode = Head;

Head = temp;

}

else

{

temp = new Node;

temp->data = val;

temp->NextNode = current;

current->PreviousNode = temp;

temp->PreviousNode = Head;

Head = temp;

}

}

void Fun::Add\_Node\_Last(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->PreviousNode = current;

temp->NextNode = NULL;

}

else

cout << endl << "!!! Create Link List First !!!" << endl;

}

void Fun::Add\_Node\_Random(int val)

{

Node \*temp = NULL, \*current = Head;

int opt = 0, counter = 0;

cout << endl << "Enter Postion to add a Node : ";

cin >> opt;

if (Head == NULL)

{

cout << endl << "Create List First !!" << endl;

}

else

{

Node \*temp1 = Head;

while (temp1 != NULL)

{

counter++;

temp1 = temp1->NextNode;

}

if (opt == 1)

{

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = NULL;

temp->PreviousNode = Head;

Head = temp;

}

else

{

temp = new Node;

temp->data = val;

current = Head;

current->PreviousNode = temp;

temp->NextNode = current;

temp->PreviousNode = Head;

Head = temp;

}

}

else if (opt == counter)

{

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->PreviousNode = current;

temp->NextNode = NULL;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter - 1; i++)

{

if (i == opt)

{

temp = new Node;

temp->data = val;

temp->NextNode = current->NextNode;

temp->PreviousNode = current;

current->NextNode->PreviousNode = temp;

current->NextNode = temp;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Delete\_Node\_First()

{

Node \*temp = NULL;

if (Head != NULL)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

Head->NextNode->PreviousNode = Head;

free(temp);

cout << endl << "First Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Last()

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << "Last Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Random()

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

int opt = 0, counter = 0;

cout << "Enter the Position of Node to Delete it : ";

cin >> opt;

while (current != NULL)

{

counter++;

current = current->NextNode;

}

if (opt == 1)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

Head->NextNode->PreviousNode = Head;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt == counter)

{

current = Head;

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter - 1; i++)

{

if (i == opt)

{

temp = new Node;

temp = current->NextNode;

current->NextNode = temp->NextNode;

temp->NextNode->PreviousNode = current;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Search\_Node\_Data(int val)

{

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

Node \*current = Head;

int counter = 1;

while (current != NULL)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

current = current->NextNode;

counter++;

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

while (temp != NULL)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Node on Start |" << endl;

cout << " | Press 2 to Add Node on Last |" << endl;

cout << " | Press 3 to Add Node Randomly |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 4 to Delete First Node |" << endl;

cout << " | Press 5 to Delete Last Node |" << "\t\t\tDOUBLY LINK LIST" << endl;

cout << " | Press 6 to Delete Random Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 7 to Search Data in Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 9 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_First(val);

break;

}

case 2:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Last(val);

break;

}

case 3:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Random(val);

break;

}

case 4:

{

List.Delete\_Node\_First();

cout << endl;

system("pause");

break;

}

case 5:

{

List.Delete\_Node\_Last();

cout << endl;

system("pause");

break;

}

case 6:

{

List.Delete\_Node\_Random();

cout << endl;

system("pause");

break;

}

case 7:

{

cout << "Enter value to search in Link List : ";

cin >> val;

List.Search\_Node\_Data(val);

cout << endl;

system("pause");

break;

}

case 9:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**A screen shot of a computer

Description automatically generated**

**Question # 6:**

**PROGRAM:**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode, \*PreviousNode;

};

class Fun

{

public:

Fun()

{

Head = NULL;

}

Node \*Head;

void Add\_Given\_Data();

void Delete\_Node\_Random();

void Display();

};

void Fun::Add\_Given\_Data()

{

Node \*temp = NULL;

int data[11] = { 12, 34, 56, 7, 5, 4, 3, 8, 9, 89, 12 };

for (int i = 0; i < 11; i++)

{

if (Head == NULL)

{

temp = new Node;

temp->data = data[i];

temp->NextNode = NULL;

temp->PreviousNode = Head;

Head = temp;

}

else

{

Node \*current = Head;

while (current->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp->data = data[i];

current->NextNode = temp;

temp->PreviousNode = current;

temp->NextNode = NULL;

}

}

}

void Fun::Delete\_Node\_Random()

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

int opt = 0, counter = 0;

cout << "Enter the Position of Node to Delete it : ";

cin >> opt;

while (current != NULL)

{

counter++;

current = current->NextNode;

}

if (opt == 1)

{

temp = new Node;

temp = Head;

Head = Head->NextNode;

Head->NextNode->PreviousNode = Head;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt == counter)

{

current = Head;

while (current->NextNode->NextNode != NULL)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode->NextNode;

current->NextNode = NULL;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter ; i++)

{

if (i == opt)

{

temp = new Node;

temp = current->NextNode;

current->NextNode = temp->NextNode;

temp->NextNode->PreviousNode = current;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

while (temp != NULL)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Data |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 2 to Delete Randomly |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 3 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Given Values in the Question Will Be Inserted Automatically !!" << endl;

List.Add\_Given\_Data();

cout << endl << "All given data is added in the list, check it from display function !" << endl;

cout << endl;

system("pause");

break;

}

case 2:

{

List.Delete\_Node\_Random();

cout << endl;

system("pause");

break;

}

case 3:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**A screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated**

**Question # 7:**

**PROGRAM:**

**NOTE: THIS CODE IS BASED ON DOUBLY LINK LIST WHICH IS CIRCULAR**

#include <iostream>

using namespace std; // BY M.ABDULLAH

struct Node

{

int data;

Node \*NextNode, \*PreviousNode;

};

class Fun

{

public:

Fun()

{

Head = NULL, Tail = NULL;

}

Node \*Head, \*Tail;

void Add\_Node\_First(int val);

void Add\_Node\_Last(int val);

void Add\_Node\_Random(int val);

void Delete\_Node\_First();

void Delete\_Node\_Last();

void Delete\_Node\_Random();

void Search\_Node\_Data(int val);

void Display();

};

void Fun::Add\_Node\_First(int val)

{

Node \*temp = NULL, \*current = NULL;

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = temp;

temp->PreviousNode = temp;

Head = temp;

Tail = temp;

}

else

{

current = Head;

temp = new Node;

temp->data = val;

temp->NextNode = current;

current->PreviousNode = temp;

if (current->NextNode == current)

{

current->NextNode = temp;

temp->PreviousNode = current;

}

else

{

while (current != Tail)

{

current = current->NextNode;

}

temp->PreviousNode = current;

current->NextNode = temp;

}

Head = temp;

}

}

void Fun::Add\_Node\_Last(int val)

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL || Tail == NULL)

{

cout << endl << "Create Link List First !!" << endl;

}

else

{

while (current != Tail)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->PreviousNode = current;

temp->NextNode = Head;

Tail = temp;

}

}

void Fun::Add\_Node\_Random(int val)

{

Node \*temp = NULL, \*current = Head;

int opt = 0, counter = 0;

cout << endl << "Enter Postion to add a Node : ";

cin >> opt;

if (Head == NULL)

{

cout << endl << "Create List First !!" << endl;

}

else

{

Node \*temp1 = Head;

if (temp1 == Tail)

{

counter++;

}

else

{

while (temp1 != Tail)

{

counter++;

temp1 = temp1->NextNode;

}

counter++;

}

if (opt == 1)

{

if (Head == NULL)

{

temp = new Node;

temp->data = val;

temp->NextNode = temp;

temp->PreviousNode = temp;

Head = temp;

Tail = temp;

}

else

{

current = Head;

temp = new Node;

temp->data = val;

temp->NextNode = current;

current->PreviousNode = temp;

if (current->NextNode == current)

{

current->NextNode = temp;

temp->PreviousNode = current;

}

else

{

while (current != Tail)

{

current = current->NextNode;

}

temp->PreviousNode = current;

current->NextNode = temp;

}

Head = temp;

}

}

else if (opt == counter)

{

while (current != Tail)

{

current = current->NextNode;

}

temp = new Node;

temp->data = val;

current->NextNode = temp;

temp->PreviousNode = current;

temp->NextNode = Head;

Tail = temp;

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter ; i++)

{

if (i == opt)

{

temp = new Node;

temp->data = val;

temp->NextNode = current->NextNode;

temp->PreviousNode = current;

current->NextNode->PreviousNode = temp;

current->NextNode = temp;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Delete\_Node\_First()

{

Node \*temp = NULL;

if (Head != NULL)

{

temp = new Node;

temp = Head;

if (temp == Tail)

{

free(temp);

Head = NULL, Tail = NULL;

}

else

{

Head = temp->NextNode;

Tail->PreviousNode = temp->NextNode;

free(temp);

}

cout << endl << "First Node has been deleted !" << endl;

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Last()

{

Node \*temp = NULL, \*current = Head;

if (Head != NULL)

{

if (current == Tail)

{

temp = new Node;

temp = Head;

free(temp);

Head = NULL, Tail = NULL;

}

else

{

while (current->NextNode != Tail)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode;

Tail = current;

current->NextNode = Head;

free(temp);

cout << endl << "Last Node has been deleted !" << endl;

}

}

else

cout << "Link List is Empty ! " << endl;

}

void Fun::Delete\_Node\_Random()

{

Node \*temp = NULL, \*current = Head;

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

int opt = 0, counter = 0;

cout << "Enter the Position of Node to Delete it : ";

cin >> opt;

Node \*temp1 = Head;

if (temp1 == Tail)

{

counter++;

}

else

{

while (temp1 != Tail)

{

counter++;

temp1 = temp1->NextNode;

}

counter++;

}

if (opt == 1)

{

temp = new Node;

temp = Head;

if (temp == Tail)

{

free(temp);

Head = NULL, Tail = NULL;

}

else

{

Head = temp->NextNode;

Tail->PreviousNode = temp->NextNode;

free(temp);

}

cout << endl << "First Node has been deleted !" << endl;

}

else if (opt == counter)

{

if (current == Tail)

{

temp = new Node;

temp = Head;

free(temp);

Head = NULL, Tail = NULL;

}

else

{

while (current->NextNode != Tail)

{

current = current->NextNode;

}

temp = new Node;

temp = current->NextNode;

Tail = current;

current->NextNode = Head;

free(temp);

cout << endl << "Last Node has been deleted !" << endl;

}

}

else if (opt > 1 && opt < counter)

{

current = Head;

for (int i = 2; i < counter ; i++)

{

if (i == opt)

{

temp = new Node;

temp = current->NextNode;

current->NextNode = temp->NextNode;

temp->NextNode->PreviousNode = current;

free(temp);

cout << endl << opt << " Node has been deleted !" << endl;

break;

}

current = current->NextNode;

}

}

}

}

void Fun::Search\_Node\_Data(int val)

{

if (Head == NULL)

{

cout << endl << "Link List is Empty !" << endl;

}

else

{

Node \*current = Head;

int counter = 1;

if (current == Tail)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

}

else

{

while (current != Tail)

{

if (current->data == val)

{

cout << endl << "Value found at " << counter << " Node" << endl;

}

current = current->NextNode;

counter++;

}

}

}

}

void Fun::Display()

{

Node \*temp = Head;

cout << endl << "DATA : ";

if (Head == NULL)

{

cout << endl << "Empty Link List " << endl;

}

else if (temp == Tail)

{

cout << " " << temp->data << " ";

}

else

{

while (temp != Tail)

{

cout << " " << temp->data << " ";

temp = temp->NextNode;

}

cout << " " << temp->data << " ";

}

cout << endl;

}

int main()

{

Fun List;

int opt = 1, val = 0;

while (opt != 0)

{

system("cls");

cout << " -----------------------------------" << endl;

cout << " | Press 1 to Add Node on Start |" << endl;

cout << " | Press 2 to Add Node on Last |" << endl;

cout << " | Press 3 to Add Node Randomly |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 4 to Delete First Node |" << endl;

cout << " | Press 5 to Delete Last Node |" << "\t\t\tDOUBLY CIRCULAR LINK LIST" << endl;

cout << " | Press 6 to Delete Random Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 7 to Search Data in Node |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 9 to Display Link List |" << endl << " |\t\t\t\t |" << endl;

cout << " | Press 0 to Exit from the system |" << endl;

cout << " -----------------------------------" << endl;

cout << endl << " Option Choosen : ";

cin >> opt;

cout << endl;

switch (opt)

{

case 1:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_First(val);

break;

}

case 2:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Last(val);

cout << endl;

system("pause");

break;

}

case 3:

{

cout << "Enter any data to Enter in List : ";

cin >> val;

List.Add\_Node\_Random(val);

cout << endl;

system("pause");

break;

}

case 4:

{

List.Delete\_Node\_First();

cout << endl;

system("pause");

break;

}

case 5:

{

List.Delete\_Node\_Last();

cout << endl;

system("pause");

break;

}

case 6:

{

List.Delete\_Node\_Random();

cout << endl;

system("pause");

break;

}

case 7:

{

cout << "Enter value to search in Link List : ";

cin >> val;

List.Search\_Node\_Data(val);

cout << endl;

system("pause");

break;

}

case 9:

{

List.Display();

cout << endl;

system("pause");

break;

}

case 0:

{

opt = 0;

cout << endl << "You have exited from Link List !" << endl;

break;

}

default:

cout << "Invalid Entry, Press any key to try again !!" << endl;

system("pause");

break;

}

}

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**