**DS LAB 10: DOUBLY LINKED LIST**

Name: Vivian Ludrick

Branch: SE Comps A Batch C

Roll No: 9914

CODE:

#include <stdio.h>

#include <stdlib.h>

*// Node structure*

typedef struct DNode

{

int data;

struct DNode \*next;

struct DNode \*prev;

} DNode;

*// Head structure*

typedef struct

{

DNode \*start;

} Head;

*// Function to create a new node*

DNode \***createNode**(int *data*)

{

DNode \*newNode = (DNode \*)**malloc**(sizeof(DNode));

if (newNode == **NULL**)

{

**printf**("Memory allocation failed\n");

**exit**(1);

}

newNode->data = *data*;

newNode->next = **NULL**;

newNode->prev = **NULL**;

return newNode;

}

*// Function to insert a node at the end of the linked list*

void **insertAtEnd**(Head \**head*, int *data*)

{

DNode \*newNode = **createNode**(*data*);

if (*head*->start == **NULL**)

{

*head*->start = newNode;

return;

}

DNode \*temp = *head*->start;

while (temp->next != **NULL**)

{

temp = temp->next;

}

temp->next = newNode;

newNode->prev = temp;

}

*// Function to insert a node at the beginning of the linked list*

void **insertAtBeginning**(Head \**head*, int *data*)

{

DNode \*newNode = **createNode**(*data*);

if (*head*->start == **NULL**)

{

*head*->start = newNode;

return;

}

newNode->next = *head*->start;

*head*->start->prev = newNode;

*head*->start = newNode;

}

*// Function to insert a node after the nth node of the linked list*

void **insertAfterNthNode**(Head \**head*, int *data*, int *n*)

{

DNode \*newNode = **createNode**(*data*);

if (*head*->start == **NULL**)

{

*head*->start = newNode;

return;

}

DNode \*temp = *head*->start;

if (*n* == 1)

{

newNode->next = *head*->start;

*head*->start->prev = newNode;

*head*->start = newNode;

return;

}

for (int i = 2; i < *n*; i++)

{

temp = temp->next;

}

if (temp == **NULL**)

{

**printf**("Invalid position\n");

return;

}

newNode->next = temp->next;

newNode->prev = temp;

if (temp->next != **NULL**)

{

temp->next->prev = newNode;

}

temp->next = newNode;

}

*// Function to delete a node from the linked list*

void **deleteNode**(Head \**head*, int *data*)

{

DNode \*temp = *head*->start;

while (temp != **NULL** && temp->data != *data*)

{

temp = temp->next;

}

if (temp == **NULL**)

{

**printf**("Node not found\n");

return;

}

if (temp->prev != **NULL**)

{

temp->prev->next = temp->next;

}

else

{

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

}

if (temp->next != **NULL**)

{

temp->next->prev = temp->prev;

}

**free**(temp);

}

*// Function to display the linked list*

void **display**(Head \**head*)

{

DNode \*temp = *head*->start;

**printf**("Linked list: ");

while (temp != **NULL**)

{

**printf**("%d ", temp->data);

temp = temp->next;

}

**printf**("\n");

}

*// Main function*

int **main**()

{

Head head;

head.start = **NULL**;

int choice, data, n;

while (1)

{

**printf**("1. Insert at end\n");

**printf**("2. Insert at beginning\n");

**printf**("3. Insert after nth node\n");

**printf**("4. Delete a node\n");

**printf**("5. Display\n");

**printf**("6. Exit\n");

**printf**("Enter your choice: ");

**scanf**("%d", &choice);

switch (choice)

{

case 1:

**printf**("Enter data: ");

**scanf**("%d", &*data*);

**insertAtEnd**(&*head*, data);

break;

case 2:

**printf**("Enter data: ");

**scanf**("%d", &*data*);

**insertAtBeginning**(&*head*, data);

break;

case 3:

**printf**("Enter data: ");

**scanf**("%d", &*data*);

**printf**("Enter position: ");

**scanf**("%d", &*n*);

**insertAfterNthNode**(&*head*, data, n);

break;

case 4:

**printf**("Enter data: ");

**scanf**("%d", &*data*);

**deleteNode**(&*head*, data);

break;

case 5:

**display**(&*head*);

break;

case 6:

**exit**(0);

default:

**printf**("Invalid choice\n");

}

}

return 0;

}

OUTPUT:



