NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)

DSA LAB SHEET NO. 5

TITLE:- IMPLEMENTATION OF DOUBLY LINKED LIST

THEORY:-

Linked List:-A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.
Doubly Linked List:- A doubly linked list (DLL) is a special type of linked list in which each node contains a pointer to the previous node as well as the next node of the linked list.

PROGRAM CODE:-

```
//program to copy and reverse list using DLL
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct DLL
{
   int data;
   struct DLL *prev;
   struct DLL *next;
};

struct DLL *first, *last;

void insertionAtBeginning(int element)
{
   struct DLL *NewNode;
   NewNode = (struct DLL *)malloc(sizeof(struct DLL));
   if (NewNode == NULL)
   {
      printf("Memory allocation failed.\n");
   }
   else
   {
      NewNode->data = element;
      NewNode->prev = NULL;
      NewNode->next = NULL;
      if (first == NULL)
      {
         first = last = NewNode;
      }
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)


```
        else
        {
            NewNode->next = first;
            first->prev = NewNode;
            first = NewNode;
        }
    }
}
void traverse()
{
    struct DLL *temp = first;
    if (first == NULL)
        printf("void or list is empty");

    else
    {
        while (temp->next != NULL)
        {
            printf("%d <<->> ", temp->data);
            temp = temp->next;
        }
        printf("%d <<->> NULL\n", temp->data);
    }
}
void reverse() // prints reverse order
{
    struct DLL *temp = last; // start temp from last

    if (first == NULL)
    {
        printf("void or list is empty\n");
    }
    else
    {
        while (temp->prev != NULL) // till reaching the first element of previous list
        {
            printf("%d <<->> ", temp->data);
            temp = temp->prev;
        }
        printf("%d <<->> NULL\n", temp->data); // for previous list first
    }
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)


```
}

struct DLL *copy(struct DLL *first)
{
    struct DLL *copyFirst = NULL;
    struct DLL *copyLast = NULL;
    struct DLL *temp = first;

    while (temp != NULL)
    {
        struct DLL *copyNode = (struct DLL *)malloc(sizeof(struct DLL));

        if (copyNode == NULL)
        {
            printf("Memory allocation failed.\n");
        }

        copyNode->data = temp->data;
        copyNode->prev = NULL;
        copyNode->next = NULL;

        if (copyFirst == NULL)
        {
            copyFirst = copyLast = copyNode;
        }
        else
        {
            copyLast->next = copyNode;
            copyNode->prev = copyLast;
            copyLast = copyNode;
        }

        temp = temp->next;
    }

    return copyFirst;
}

int main()
{
    insertionAtBeginning(40);
    insertionAtBeginning(30);
```

```
insertionAtBeginning(20);
insertionAtBeginning(10);

printf("Original List: ");
traverse();

printf("Reversed List: ");
reverse();

struct DLL *copyList = copy(first);

printf("Copied List: ");
traverse();
/*if we comment out the copy function then the program will also run as it is
but the traverse then will print the previous list
so to know that copy function is working properly
we compare the datas of the copied list from copy function with our original list
that will prove copy function is working or not.*/

struct DLL *originalTemp = first;
struct DLL *copyTemp = copyList;

while (originalTemp != NULL && copyTemp != NULL)
{
    if (originalTemp->data != copyTemp->data)
    {
        printf("copied data is incorrect.\n");
        break;
    }

    originalTemp = originalTemp->next;
    copyTemp = copyTemp->next;
}

if (originalTemp != NULL || copyTemp != NULL)
{
    printf("copy function failed to create a complete copy of the list.\n");
}
else
{
    printf("The copy function is working.\n");
}
```

NAME:- SAGAR ADHIKARI
ROLL NO:- PUR078BEI034
FACULTY:- ELECTRONICS(BEI)


```
    return 0;
}
```

OUTPUT:-

```
Original List: 10 <<->> 20 <<->> 30 <<->> 40 <<->> NULL
Reversed List: 40 <<->> 30 <<->> 20 <<->> 10 <<->> NULL
Copied List: 10 <<->> 20 <<->> 30 <<->> 40 <<->> NULL
The copy function is working.
```