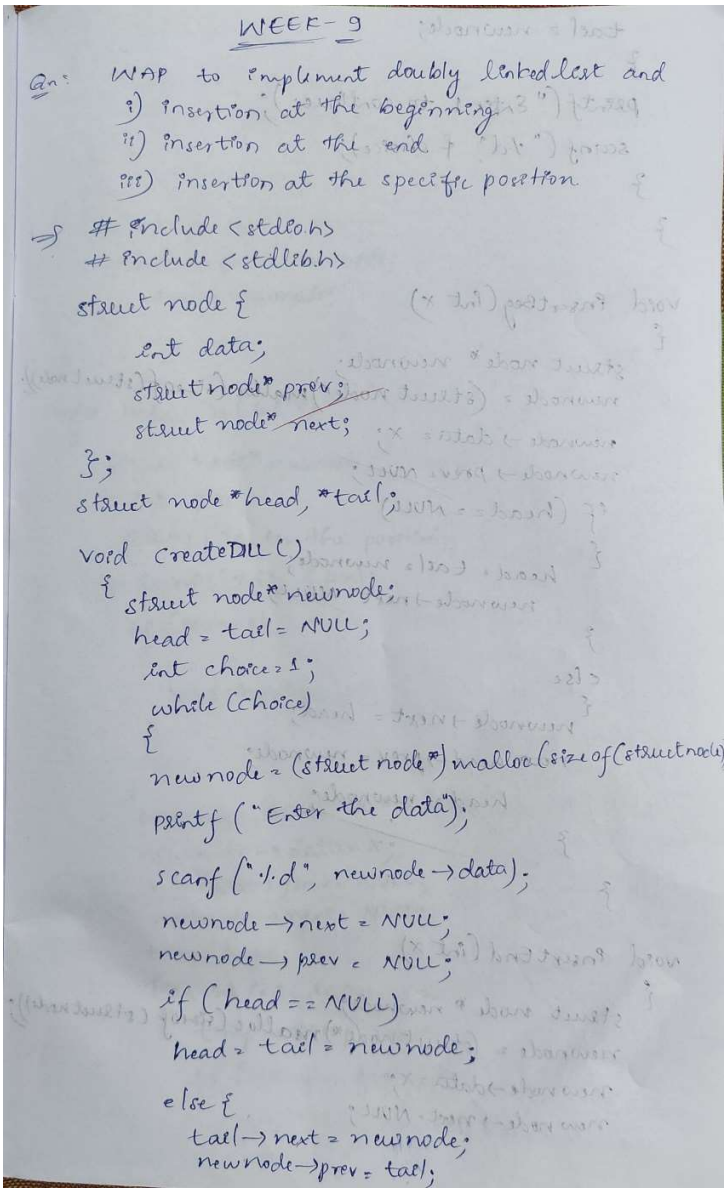
Program 8

Write a Program to Implement doubly link list with primitive operations
a) Create a doubly linked list.
b) Insert a new node at the beginning.
c) Insert the node based on a specific location
d) Insert a new node at the end.
e) Display the contents of the list

Observation:



WEEK-9

Q: WAP to implement doubly linked list and
i) insertion at the beginning
ii) insertion at the end
iii) insertion at the specific position

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* prev;
    struct node* next;
};
struct node *head, *tail;
void createDLL()
{
    struct node* newnode;
    head = tail = NULL;
    int choice = 1;
    while (choice)
    {
    newnode = (struct node*) malloc (sizeof(struct node));
    printf ("Enter the data");
    scanf ("%d", newnode->data);
    newnode->next = NULL;
    newnode->prev = NULL;
    if (head == NULL)
     head = tail = newnode;
    else {
      tail->next = newnode;
      newnode->prev = tail;
      tail = newnode;
    }
    printf ("Enter 1 to continue:");
    scanf ("%d", &choice);
    }
}

void InsertBeg (int x)
{
    struct node* newnode;
    newnode = (struct node*) malloc (sizeof(struct
    newnode->data = x;
    newnode->prev = NULL;
    if (head == NULL)
    {
      head = tail = newnode;
      newnode->next = NULL;
    }
    else
    {
      newnode->next = head;
      head->prev = newnode;
      head = newnode;
    }
}

void InsertEnd (int x)
{
    struct node* newnode;
    newnode = (struct node*) malloc (sizeof(struct
    newnode->data = x;
    newnode->next = NULL;
```

```
if (head == NULL)
{
    head = tail = newnode;
    newnode->prev = NULL;
}
else
{ newnode->prev = tail;
  tail->next = newnode;
    tail = newnode;
}
}
void insertPos (int x)
{
  struct node* newnode, *temp;
  int pos;
  printf ("Enter the position");
  scanf ("%d", &pos);
  if (pos==1)
  {
    void insertBeg(x);
  }
  else
  {
  newnode = (struct node*) malloc (sizeof(struct node));
  newnode -> data = x;
  newnode -> prev = NULL;
  newnode -> next = NULL;
  temp = head;
  for (int i=1, i<pos-1; i++)
  { temp = temp->next;
    if (temp == tail->next)
    { printf("There are less than %d nodes", pos);
      return;
  } }
```

```
newnode->prev = temp;
newnode->next = temp->next;
temp->next = newnode;
newnode->next->prev = newnode;
}
}
```

Seen

Output:

```
Enter the data: 1
enter 1 to continue: 1
Enter the data: 2
enter 1 to continue: 1
Enter the data: 3
enter 1 to continue: 2
Menu:
1. Insert at Beginning
2. Insert at specific position
3. Insert at End
4. Display list
5. Exit
Enter your choice: 1
Enter value to insert at the beginning: 4
Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display list
5. Exit.
Enter your choice: 4
4 <-> 1 <-> 2 <-> 3 <-> NULL
```

```
Menu:
1. Insert at Beginning
2. Insert at specific Position
3. Insert at End
4. Display list
5. Exit
Enter your choice: 2
Enter value to insert: 5
Enter the position: 2
Menu:
1. Insert at Beginning
2. Insert at specific position
3. Insert at End
4. Display list
5. Exit
Enter your choice: 4
4 <-> 5 <-> 1 <-> 2 <-> 3 <-> NULL

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display list
5. Exit
Enter your choice: 3
Enter value to insert at the end: 6
Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display list
5. Exit.
Enter your choice: 4
4 <-> 5 <-> 1 <-> 2 <-> 3 <-> 6 <-> NULL
```

Seen

Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head, *tail;
void createDLL(){
    struct node *newnode;
    head = tail = NULL;
    int choice = 1;
    while(choice==1){
        newnode = (struct node *)malloc(sizeof(struct node));
        printf("Enter the data:");
        scanf("%d", &newnode->data);
        newnode->next = NULL;
        newnode->prev = NULL;
        if(head == NULL){
        head = tail = newnode;
        }
        else{
        tail->next = newnode;
        newnode->prev = tail;
        tail = newnode;
        }
        printf("enter 1 to continue:");
        scanf("%d", &choice);
    }
}
void insertBeg(int x){
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = x;
    newnode->prev = NULL;
    if(head == NULL){
    head = tail = newnode;
    newnode->next = NULL;
    }
    else{
    newnode->next = head;
    head->prev = newnode;
    head = newnode;
    }
}
void insertEnd(int x){
    struct node *newnode;
    newnode = (struct node *)malloc(sizeof(struct node));
```

```
    newnode->data = x;
    newnode->next = NULL;
    if(head == NULL){
    head = tail = newnode;
    newnode->prev = NULL;
    }
    else{
    newnode->prev = tail;
    tail->next = newnode;
    tail = newnode;
    }
}
void insertPos(int x){
    struct node *newnode, *temp;
    int pos;
    printf( "Enter the position:");
    scanf("%d", &pos);
    if(pos == 1){
        insertBeg(x);
    }
    else{
        newnode = (struct node *)malloc(sizeof(struct node));
        newnode->data = x;
        newnode->next = NULL;
        newnode->prev = NULL;
        temp = head;
        for(int i = 1; i < pos-1 ; i++){
            temp=temp->next;
            if(temp == tail->next){
                printf("There are less than %d nodes\n",pos);
                return;
            }
        }
        newnode->prev = temp;
        newnode->next = temp->next;
        temp->next = newnode;
        newnode->next->prev = newnode;
    }
}
void display(){
    struct node *temp;
    if(head == NULL){
        printf("list is empty\n");
    }
    else{
        temp = head;
        while(temp != NULL){
            printf("%d<->", temp->data);
            temp = temp->next;
        }
```

```
        printf("NULL\n");
    }
}
int main() {
 createDLL();
 int choice, value, position;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at Specific Position\n");
        printf("3. Insert at End\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert at the beginning: ");
                scanf("%d", &value);
                insertBeg(value);
                break;

            case 2:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                insertPos(value);
                break;

            case 3:
                printf("Enter value to insert at the end: ");
                scanf("%d", &value);
                insertEnd(value);
                break;

            case 4:
                display();
                break;

            case 5:
                printf("Exiting...\n");
                return 0;

            default:
                printf("Invalid choice, please try again.\n");
        }
    }
    return 0;
}
```

Output:

```
Enter the data:1
enter 1 to continue:1
Enter the data:2
enter 1 to continue:1
Enter the data:3
enter 1 to continue:2

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 1
Enter value to insert at the beginning: 4

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 4
4<->1<->2<->3<->NULL

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 2
Enter value to insert: 5
Enter the position:2

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 4
4<->5<->1<->2<->3<->NULL

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 3
```

```
Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 3
Enter value to insert at the end: 6

Menu:
1. Insert at Beginning
2. Insert at Specific Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 4
4<->5<->1<->2<->3<->6<->NULL
```