Program 7

Write A Program to Implement doubly link list with primitive operations

a) Create a doubly linked list.

b) Insert a new node to the left of the node.

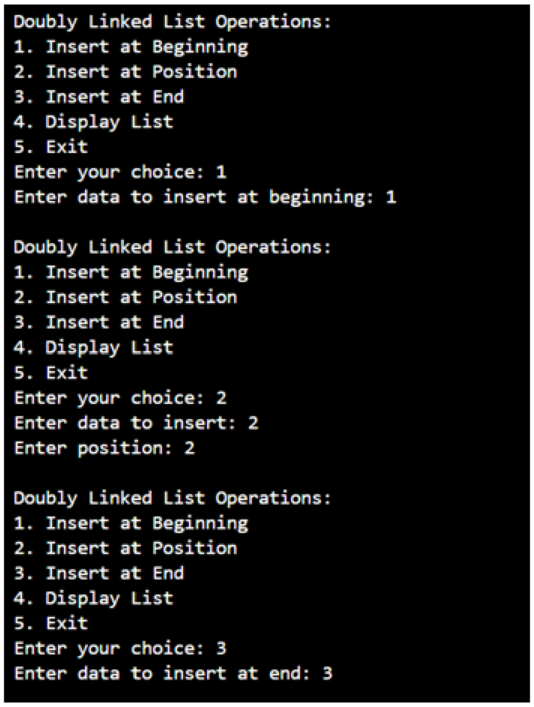
c) Delete the node based on a specific value

d) Display the contents of the list

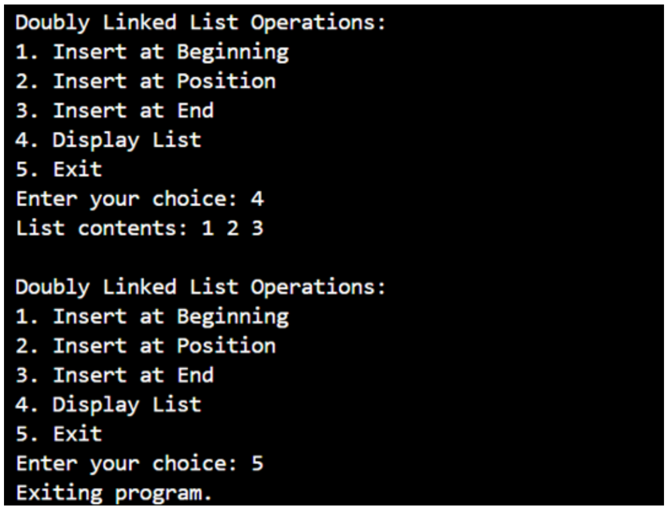
Code:

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

void insertAtBeginning(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
```

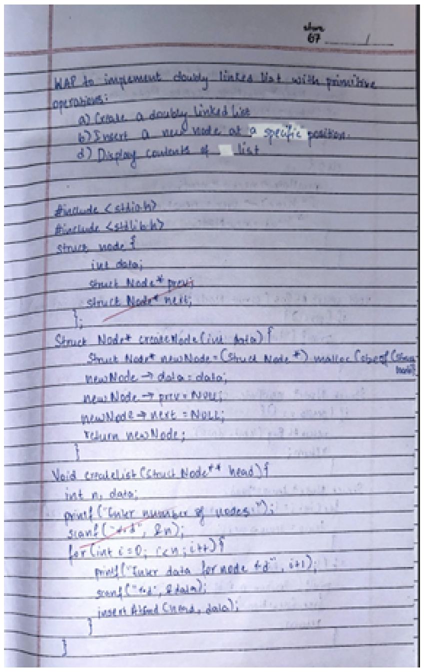
```
void insertAtPosition(struct Node** head, int data, int position) {
    if (position < 1) {
        printf("Invalid position!\n");
        return;
    }

    struct Node* newNode = createNode(data);
    if (position == 1) {
        insertAtBeginning(head, data);
        return;
    }

    struct Node* temp = *head;
    for (int i = 1; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Position out of bounds!\n");
        free(newNode);
        return;
    }

    newNode->next = temp->next;
    newNode->prev = temp;
    if (temp->next != NULL) {
        temp->next->prev = newNode;
    }
    temp->next = newNode;
}

void insertAtEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }

    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

void displayList(struct Node* head) {
    if (head == NULL) {
        printf("List is empty!\n");
        return;
    }

    struct Node* temp = head;
    printf("List contents: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Node* head = NULL;
    int choice, data, position;

    while (1) {
        printf("\nDoubly Linked List Operations:\n");
        printf("1. Insert at Beginning\n");
        printf("2. Insert at 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 data to insert at beginning: ");
                scanf("%d", &data);
                insertAtBeginning(&head, data);
                break;
            case 2:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &position);
                insertAtPosition(&head, data, position);
                break;
            case 3:
                printf("Enter data to insert at end: ");
                scanf("%d", &data);
                insertAtEnd(&head, data);
                break;
            case 4:
                displayList(head);
                break;
            case 5:
                printf("Exiting program.\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }
    return 0;
}
```



```
Doubly Linked List Operations:
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 1
Enter data to insert at beginning: 1

Doubly Linked List Operations:
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 2
Enter data to insert: 2
Enter position: 2

Doubly Linked List Operations:
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 3
Enter data to insert at end: 3
```



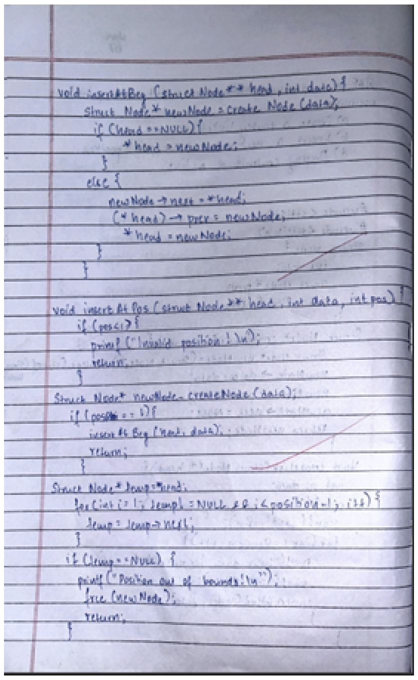
```
Doubly Linked List Operations:
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 4
List contents: 1 2 3

Doubly Linked List Operations:
1. Insert at Beginning
2. Insert at Position
3. Insert at End
4. Display List
5. Exit
Enter your choice: 5
Exiting program.
```



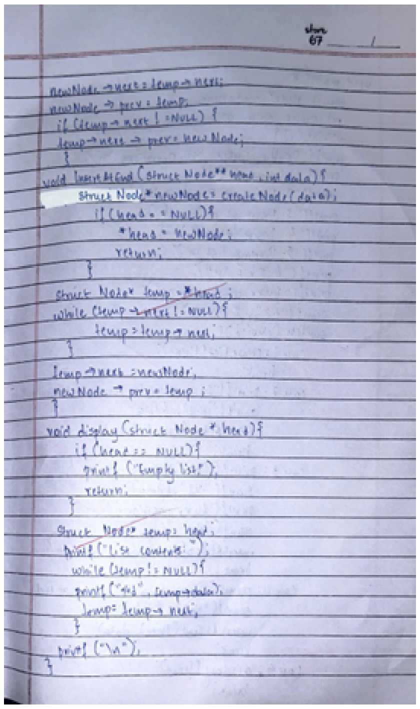
67

WAP to implement doubly linked list with primitive operations:
a) Create a doubly linked list
b) Insert a new node at a specific position
d) Display contents of list

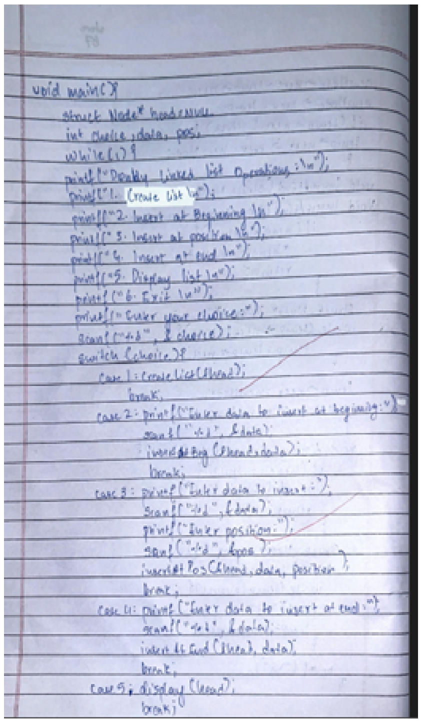
```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct Node* prev;
    struct Node* next;
 };
Struct Node* createNode(int data) {
    Struct Node* newNode = (Struct Node *) malloc (sizeof(Struct Node));
    newNode -> data = data;
    newNode -> prev = NULL;
    newNode -> next = NULL;
    return newNode;
 }
Void createList(Struct Node** head) {
  int n, data;
  printf("Enter number of nodes:");
  scanf("%d", &n);
  for(int i=0; i<n; i++) {
      printf("Enter data for node %d", i+1);
      scanf("%d", &data);
      insertAtEnd(head, data);
   }
 }
```



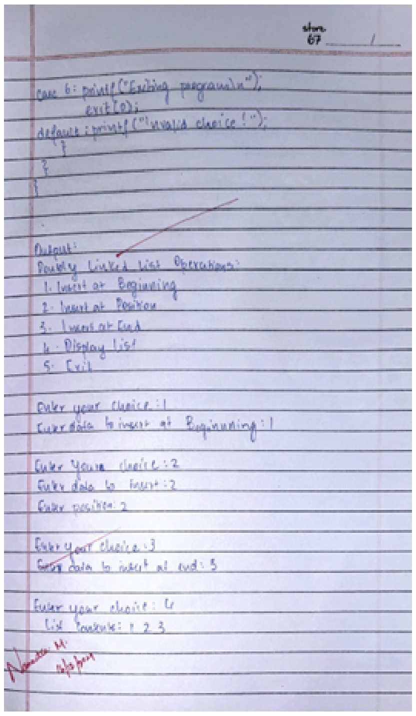
```
void insertAtBeg (struct Node** head, int data) {
    Struct Node* newNode = create Node (data);
     if (*head == NULL) {
        *head = newNode;
      }
    else {
       newNode -> next = *head;
       (*head) -> prev = newNode;
        *head = newNode;
     }
  }

void insertAtPos (struct Node** head, int data, int pos) {
    if (pos<1) {
      printf("Invalid position!\n");
      return;
    }
Struck Node* newNode = createNode (data);
   if (pos == 1) {
      insertAtBeg (head, data);
      return;
    }
Struct Node* temp = *head;
   for(int i=1; temp != NULL && i<position-1; i++) {
     temp = temp -> next;
   }
  if (temp == NULL) {
    printf("Position out of bounds!\n");
     free (newNode);
     return;
  }
```



```
newNode → next = temp → next;
newNode → prev = temp;
 if (temp → next != NULL) {
 temp → next → prev = newNode;
  }
void insertAtEnd (struct Node** head, int data) {
      struct Node* newNode = createNode (data);
        if (head = = NULL) {
            *head = newNode;
            return;
        }
 struct Node* temp = *head;
 while (temp → next != NULL) {
        temp = temp → next;
  }
 temp → next = newNode;
 newNode → prev = temp;
 }
void display (struct Node * head) {
     if (head == NULL) {
      printf ("Empty list");
      return;
    }
  struct Node* temp = head;
   printf ("List contents: ");
    while (temp != NULL) {
     printf ("%d", temp → data);
      temp = temp → next;
    }
  printf ("\n");
 }
```



```
void main() {
    struct Node* head = NULL;
    int choice, data, pos;
    while (1) {
    printf ("Doubly Linked list Operations :\n");
    printf ("1. Create list \n");
     printf ("2. Insert at Beginning \n");
     printf (" 3. Insert at position \n");
     printf (" 4. Insert at end \n");
     printf ("5. Display list \n");
      printf ("6. Exit \n");
      printf ("Enter your choice :");
      scanf ("%d", & choice);
      switch (choice) {
        case 1 : createList(&head);
              break;
        case 2 : printf ("Enter data to insert at beginning:");
                scanf ("%d", &data);
                insertAtBeg (&head, data);
                 break;
       case 3 : printf ("Enter data to insert :");
                scanf ("%d", &data);
                printf ("Enter position :");
                scanf ("%d", &pos);
                insertAtPos (&head, data, position );
                break;
        case 4 : printf ("Enter data to insert at end :");
                scanf ("%d", &data);
                insertAtEnd (&head, data);
                break;
       case 5 : display (head);
                break;
```



```
case 6 : printf ("Exiting program\n");
         exit(0);
default : printf ("Invalid choice !");
   }
  }
}
```

Output:

```
Doubly Linked List Operations:
 1. Insert at Beginning
 2. Insert at Position
 3. Insert at End
 4. Display List
 5. Exit

Enter your choice : 1
Enter data to insert at Beginning : 1

Enter your choice : 2
Enter data to insert : 2
Enter position : 2

Enter your choice : 3
Enter data to insert at end : 3

Enter your choice : 4
 List contents : 1 2 3
```

[illegible]