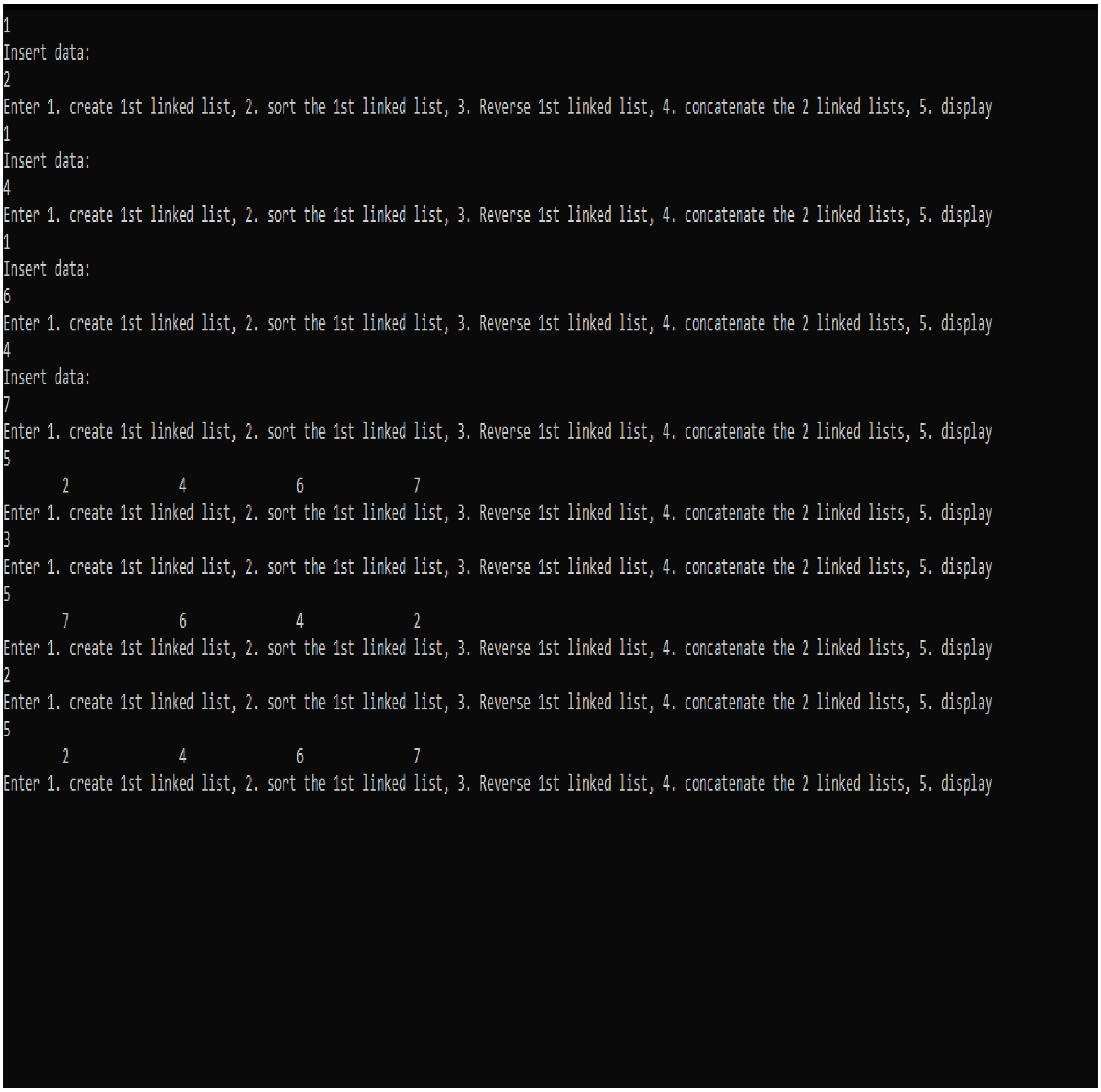
1. **6a) WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.**

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

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

struct Node *head1, *newnode, *head2, *temp1, *temp2, *prev, *n, *temp, *current, *index;

void create1() {
   newnode = (struct Node*)malloc(sizeof(struct Node));
   printf("Insert data:\n");
   scanf("%d", &newnode->data);
   if (head1 == NULL) {
      head1 = temp1 = newnode;
      temp1->next = NULL;
   } else {
      temp1->next = newnode;
      temp1 = newnode;
      temp1->next = NULL;
   }
}

void create2() {
   newnode = (struct Node*)malloc(sizeof(struct Node));
   printf("Insert data:\n");
   scanf("%d", &newnode->data);
   if (head2 == NULL) {
      head2 = temp2 = newnode;
      temp2->next = NULL;
   } else {
      temp2->next = newnode;
      temp2 = newnode;
      temp2->next = NULL;
   }
}

void concat() {
   create2();
   if (head1 == NULL) {
        head1 = head2;
    } else {
        temp1 = head1;
        while (temp1->next != NULL) {
            temp1 = temp1->next;
        }
        temp1->next = head2;
    }
}

void reverse() {
    prev = NULL;
    temp = head1;
    while (temp != NULL) {
        n = temp->next;
        temp->next = prev;
        prev = temp;
        temp = n;
    }
    head1 = prev;
}

void sort() {
    current = head1;
    int temp;
    while (current != NULL) {
        index = current->next;
        while (index != NULL) {
            if (current->data > index->data) {
                temp = current->data;
                current->data = index->data;
                index->data = temp;
            }
            index = index->next;
        }
        current = current->next;
    }
}

void display() {
    temp1 = head1;
    while (temp1 != NULL) {
        printf("\t%d\t", temp1->data);
        temp1 = temp1->next;
    }
    printf("\n");
}
```

```
int main() {
    head1 = NULL;
    head2 = NULL;
    index = NULL;
    while (1) {
        printf("Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate
the 2 linked lists, 5. display\n");
        int choice;
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                create1();
                break;
            case 2:
                sort();
                break;
            case 3:
                reverse();
                break;
            case 4:
                concat();
                break;
            case 5:
                display();
                break;
            default:
                exit(1);
        }
    }
    return 0;
}
```

OUTPUT:



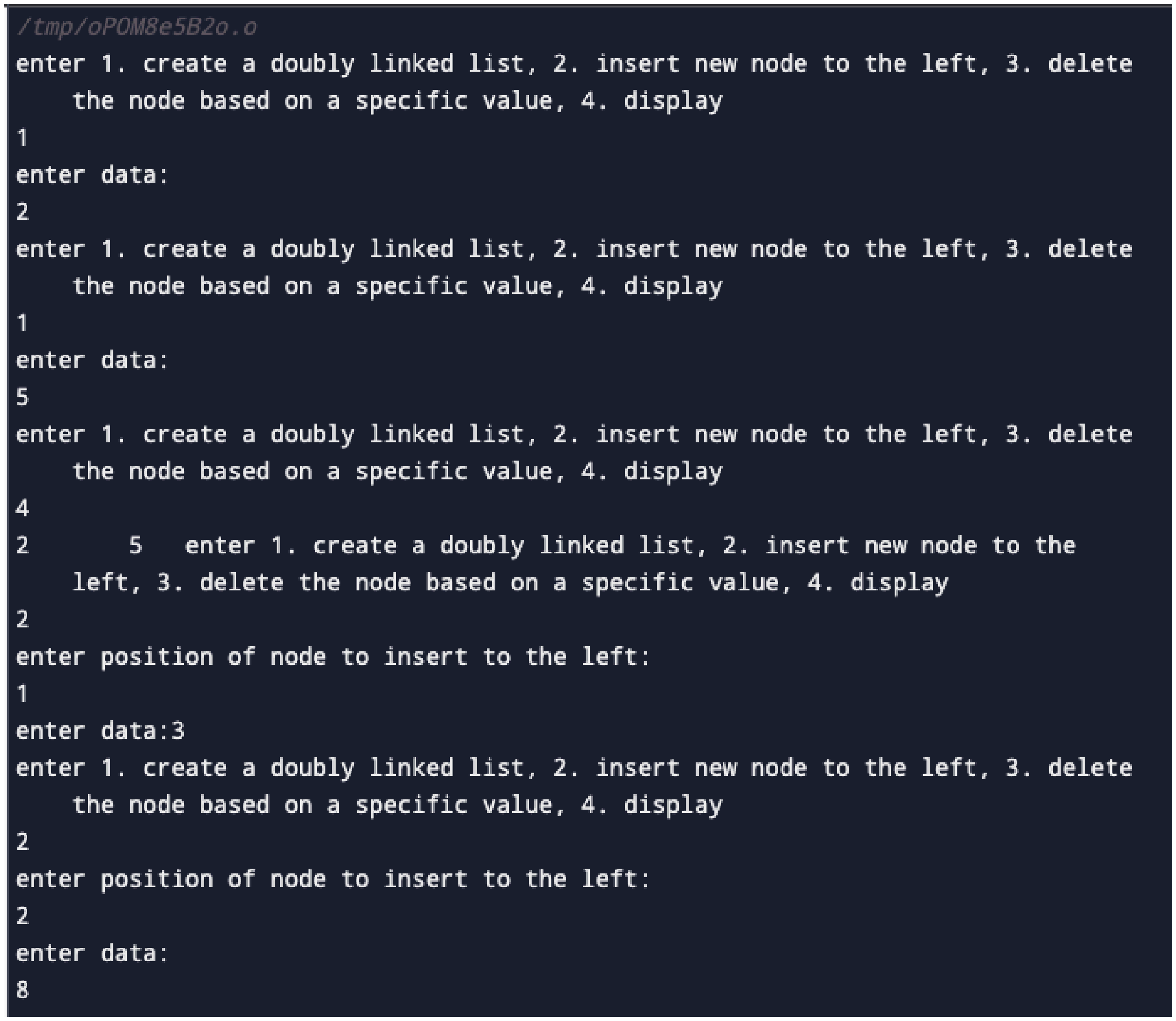
```
1
Insert data:
2
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
1
Insert data:
4
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
1
Insert data:
6
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
4
Insert data:
7
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
5
        2               4               6               7
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
3
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
5
        7               6               4               2
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
2
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
5
        2               4               6               7
Enter 1. create 1st linked list, 2. sort the 1st linked list, 3. Reverse 1st linked list, 4. concatenate the 2 linked lists, 5. display
```

2. **8)WAP to Implement doubly link list with primitive operations**
I.Create a doubly linked list.
II. Insert a new node to the left of the node.

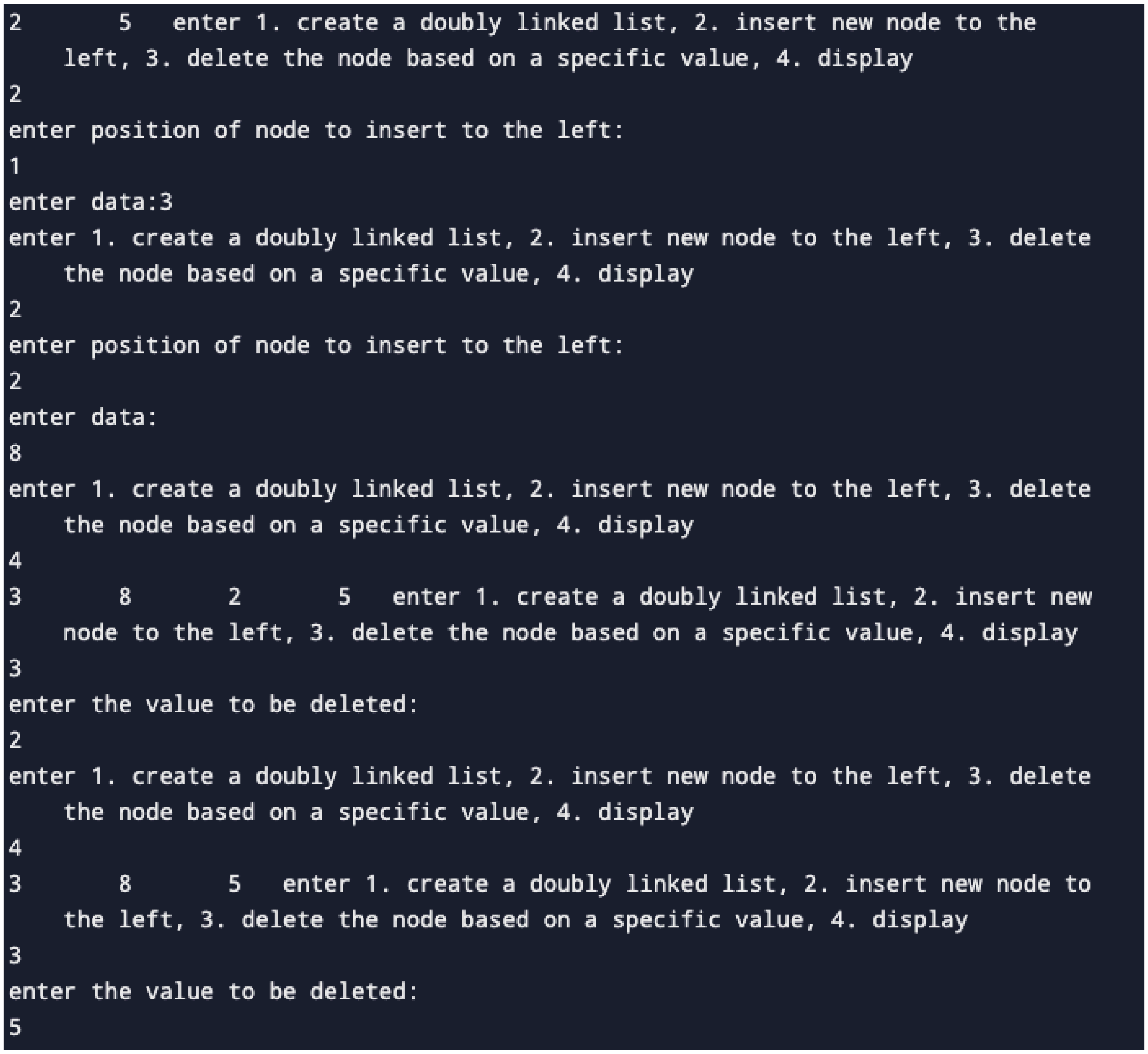
III. Delete the node based on a specific value
IV. Display the contents of the list

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

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

struct node *head, *temp, *p, *f, *ptr,*newnode;

void create(){
   newnode=(struct node*)malloc(sizeof(struct node));
   printf("enter data:\n");
   scanf("%d",&newnode->data);
   if(head==NULL){
      head=temp=newnode;
      temp->prev=NULL;
      temp->next=NULL;
   }
   else{
      temp->next=newnode;
      newnode->prev=temp;
      temp=temp->next;
   }
}

void insertLeft(){
   temp=head;
   int pos;
   printf("enter position of node to insert to the left:\n");
   scanf("%d",&pos);
   int i=1;
   if(pos==1){
      newnode=(struct node*)malloc(sizeof(struct node));
      printf("enter data:");
      scanf("%d",&newnode->data);
      newnode->next=temp;
      head=newnode;
      newnode->prev=NULL;
   }
   else{
      while(i<pos){
          p=temp;
          temp=temp->next;
          i++;
      }
      newnode=(struct node*)malloc(sizeof(struct node));
      printf("enter data:\n");
      scanf("%d",&newnode->data);
      newnode->next=temp;
      p->next=newnode;
      newnode->prev=p;
   }
}

void delete(){
   temp=head;
   f=temp;
   int val;
   printf("enter the value to be deleted:\n");
   scanf("%d",&val);
   while(temp!=NULL){
      if(val==temp->data){
          if(temp==head){
              temp=temp->next;
              head=temp;
              f->next=NULL;
              free(f);
          }
          else if(temp->next==NULL){
              f=temp;
              temp->prev=NULL;
              free(f);
          }
          else{
              f->next=temp->next;
              temp->next->prev=f;
              temp->next=NULL;
              temp->prev=NULL;
              ptr=temp;
              free(ptr);
          }
      }
      else{
          f=temp;
          temp=temp->next;
      }
   }
}

void display(){
   temp=head;
   while(temp!=NULL){
      printf("\t%d\t",temp->data);
      temp=temp->next;
   }
}

void main(){
   head=NULL;
   while(1){
      printf("enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete the node based on a
specific value, 4. display\n");

      int choice;
      scanf("%d",&choice);
      switch(choice){
          case 1: create();
                  break;
          case 2: insertLeft();
                  break;
          case 3: delete();
                  break;
          case 4: display();
                  break;
          default: exit(1);
      }
   }
}
```

OUTPUT:



```
/tmp/oPOM8e5B2o.o
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
1
enter data:
2
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
1
enter data:
5
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
4
2       5   enter 1. create a doubly linked list, 2. insert new node to the
    left, 3. delete the node based on a specific value, 4. display
2
enter position of node to insert to the left:
1
enter data:3
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
2
enter position of node to insert to the left:
2
enter data:
8
```



```
2       5   enter 1. create a doubly linked list, 2. insert new node to the
    left, 3. delete the node based on a specific value, 4. display
2
enter position of node to insert to the left:
1
enter data:3
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
2
enter position of node to insert to the left:
2
enter data:
8
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
4
3       8       2       5   enter 1. create a doubly linked list, 2. insert new
    node to the left, 3. delete the node based on a specific value, 4. display
3
enter the value to be deleted:
2
enter 1. create a doubly linked list, 2. insert new node to the left, 3. delete
    the node based on a specific value, 4. display
4
3       8       5   enter 1. create a doubly linked list, 2. insert new node to
    the left, 3. delete the node based on a specific value, 4. display
3
enter the value to be deleted:
5
```

SANJANA SHETTY- 1BM22CS238