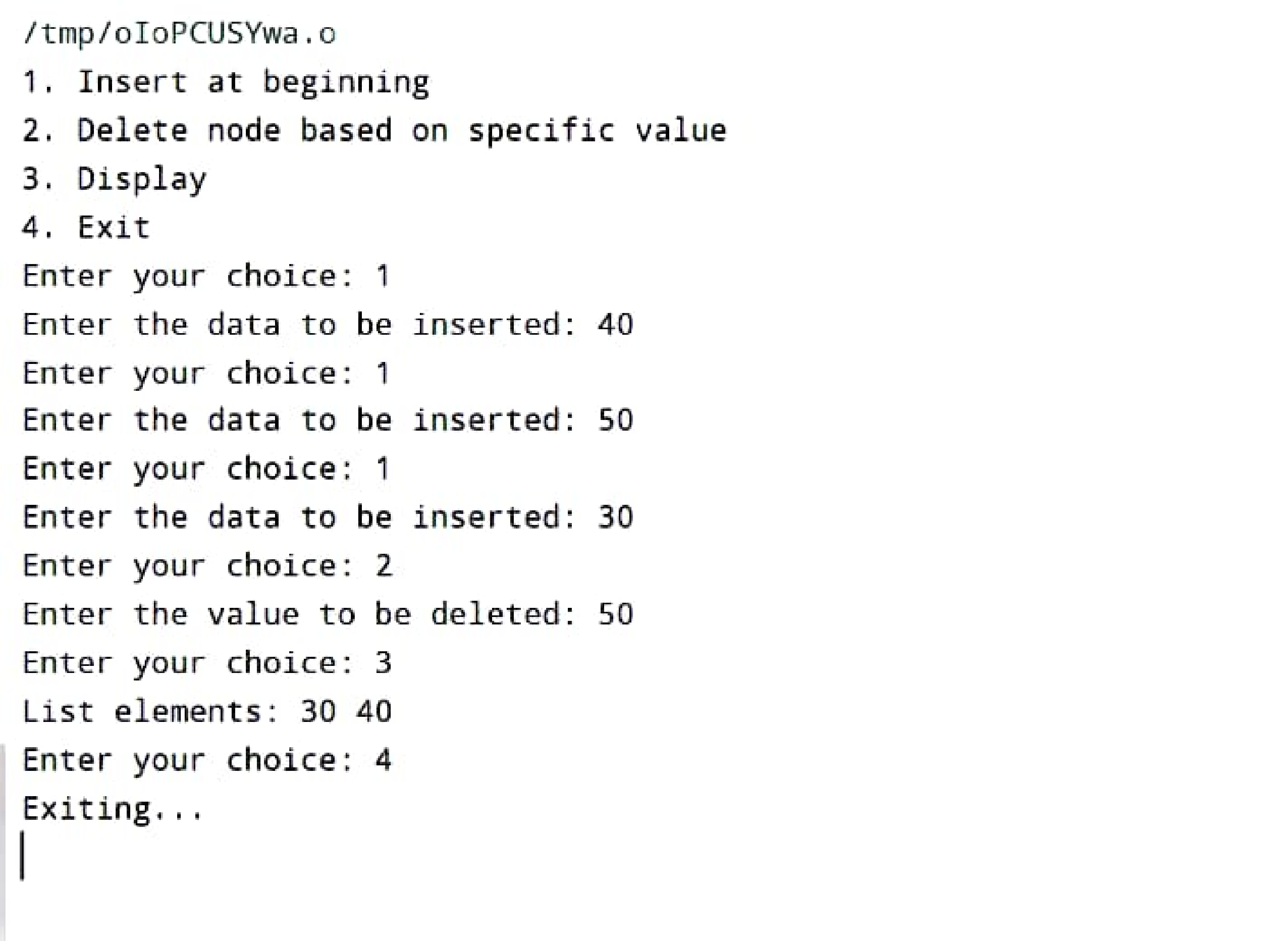
```c
#include <stdio.h>
#include <stdlib.h>

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

struct Node* create(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

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

void deleteNode(struct Node** head, int value) {
    if (*head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = *head;
    while (temp != NULL && temp->data != value) {
        temp = temp->next;
    }
    if (temp == NULL) {
        printf("Value not found in the list\n");
        return;
    }
    }
    if (temp == NULL) {
        printf("Value not found in the list\n");
        return;
    }
    if (temp->prev == NULL) {
        *head = temp->next;
        if (temp->next != NULL) {
            temp->next->prev = NULL;
        }
    } else {
        temp->prev->next = temp->next;
        if (temp->next != NULL) {
            temp->next->prev = temp->prev;
        }
    }
    free(temp);
}

void display(struct Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("List elements: ");
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}
```

```c
int main() {
    struct Node* head = NULL;
    int choice, data;
    printf("1. Insert at beginning\n");
    printf("2. Delete node based on specific value\n");
    printf("3. Display\n");
    printf("4. Exit\n");
    while (1) {
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter the data to be inserted: ");
                scanf("%d", &data);
                insertAtBeginning(&head, data);
                break;
            case 2:
                printf("Enter the value to be deleted: ");
                scanf("%d", &data);
                deleteNode(&head, data);
                break;
            case 3:
                display(head);
                break;
            case 4:
                printf("Exiting...\n");
                exit(EXIT_SUCCESS);
            default:
                printf("Invalid choice\n");
        }
    }
    return 0;
}
```

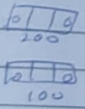


```
/tmp/oIoPCUSYwa.o
1. Insert at beginning
2. Delete node based on specific value
3. Display
4. Exit
Enter your choice: 1
Enter the data to be inserted: 40
Enter your choice: 1
Enter the data to be inserted: 50
Enter your choice: 1
Enter the data to be inserted: 30
Enter your choice: 2
Enter the value to be deleted: 50
Enter your choice: 3
List elements: 30 40
Enter your choice: 4
Exiting...
```

1. WAP to implement doubly link list with primitive operations

a) Create a doubly linked list
b) insert a newnode to left of node
c) delete the node based on a specific value

```
struct node
{
  int data;
  struct node * next;
  struct node * prev;
};
struct node * create (int value)
{
  struct node * newnode = (~~struct~~ node *)malloc(sizeof(struct node));
  newnode → next = 0.
  ~~newnode → prev = 0;~~
  newnode → data = value;
  return newnode;
}
struct node * insertatbeg (struct node * head, int value)
{
  struct node * newnode = create(value);
  ~~newnode → next = head~~
  ~~struct node * temp = newnode~~;
    ~~temp → next~~ =
    head → prev = newnode
    newnode → next = head
    newnode → prev = null.
    return newnode;
}
```



```
struct node * delete (int value)
{
        struct node * temp = head.
        while (temp → data != value)
          {
             temp = temp → next;
          }
         temp → prev → next = temp → next
         temp → next → prev = temp → prev
         free(temp);
}
```

10 20 30
100 200 300

~~Output~~

```
~~struct~~ void display()
   {
     struct node * temp = head;
     while (temp != NULL)
     {
       printf("%d", temp → data);
       temp = temp → next;
     }
   }
```

main f^n

```
int main(){
  struct node *head =NULL;
  int choice, position, value;
  choice=0;
  printf("Enter 1. To ~~[illegible]~~ insert a node to
        the left 2. To delete a node 3. to display
                                    4. exit");
  while (choice != 4)
  {
    printf("Enter choice\n");
    scanf("%d", &choice);
    switch(choice){
     case 1:  printf("~~Insert a~~ Enter data ~~node at beg~~ =");
              scanf("%d",&value);
         head= insertatbeg(head, value);
              break;
     case 2:  printf("Enter value to be deleted");
              scanf("%d",&value);
              delete(value);
              break;
     case 3: display();
  }
}
```

Output:

```
Enter 1. To insert a node to the left 2. To delete
   a node 3. To display 4. Exit
Enter choice
1
Enter data 10
Enter choice
1
Enter data 20
Enter choice
1
Enter data 30
Enter choice
3
30 → 20 → 10
Enter choice
2
Enter value to be deleted 20
Enter choice
3
30 → 10
```

[illegible] 05.02.24