Report on

"Data Structures"

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering in the course of ***Data Structures*** ***(***19CS3PCDST***)***

Submitted by

Dhruva M
(1BM19CS049)

Under the Guidance of
Dr. Kayarvizhy N.
Associate Professor
Department of CSE

Department of Computer Science and Engineering
BMS College of Engineering
P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019
2020-2021

B M S COLLEGE OF ENGINEERING

P.O. Box No: 1908 Bull Temple Road Bangalore-560019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ASSESSMENT

Report on **Data Structures (19CS3PCDST),** "Advanced Algorithm assignment" has been successfully completed by **Dhruva M** at B.M.S College of Engineering in partial fulfillment of the requirements for the 3rd Semester, degree in Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during academic year 2020-2021.

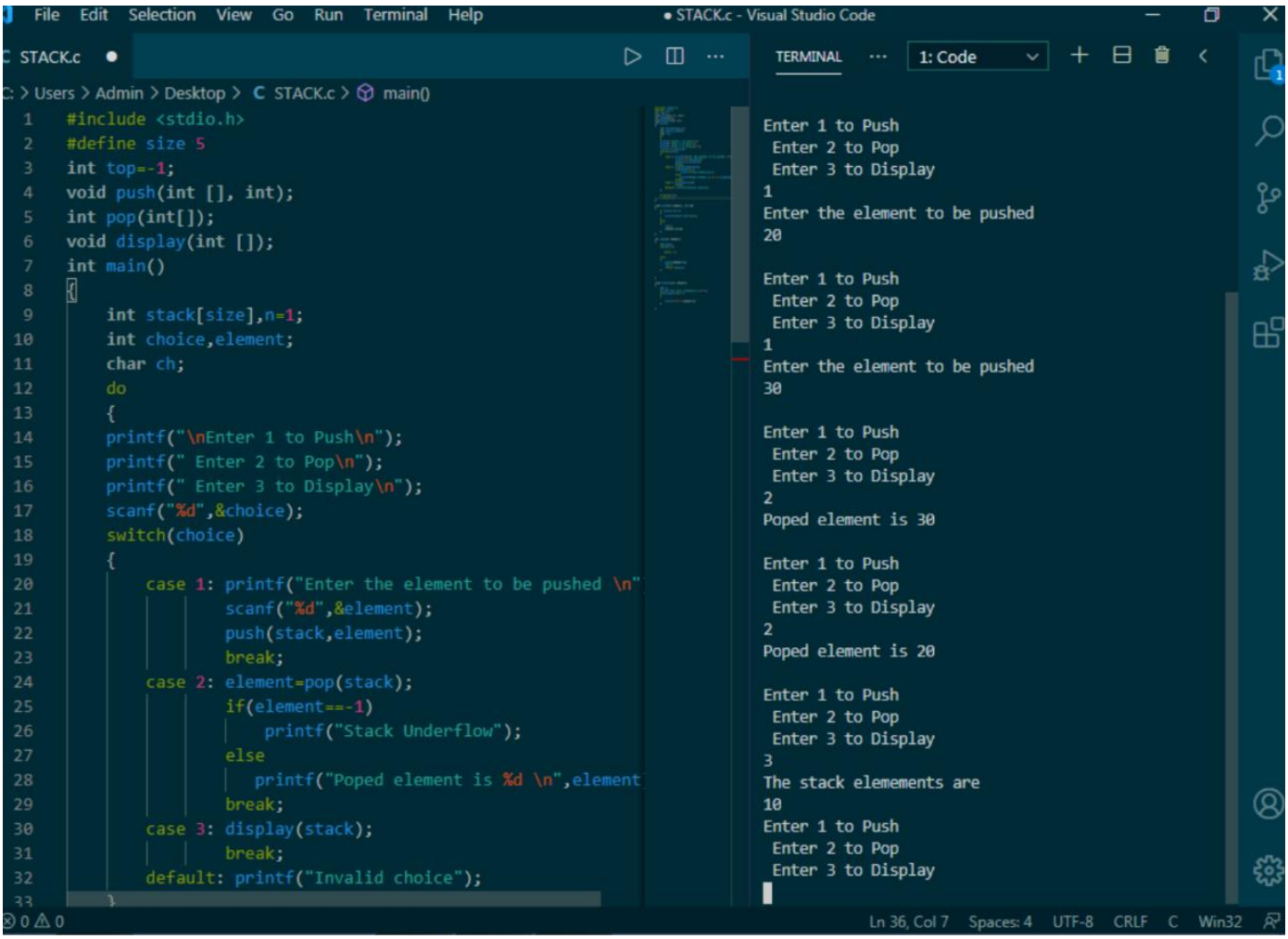
Dr. Kayarvizhy N.
Associate Professor
Department of Computer science

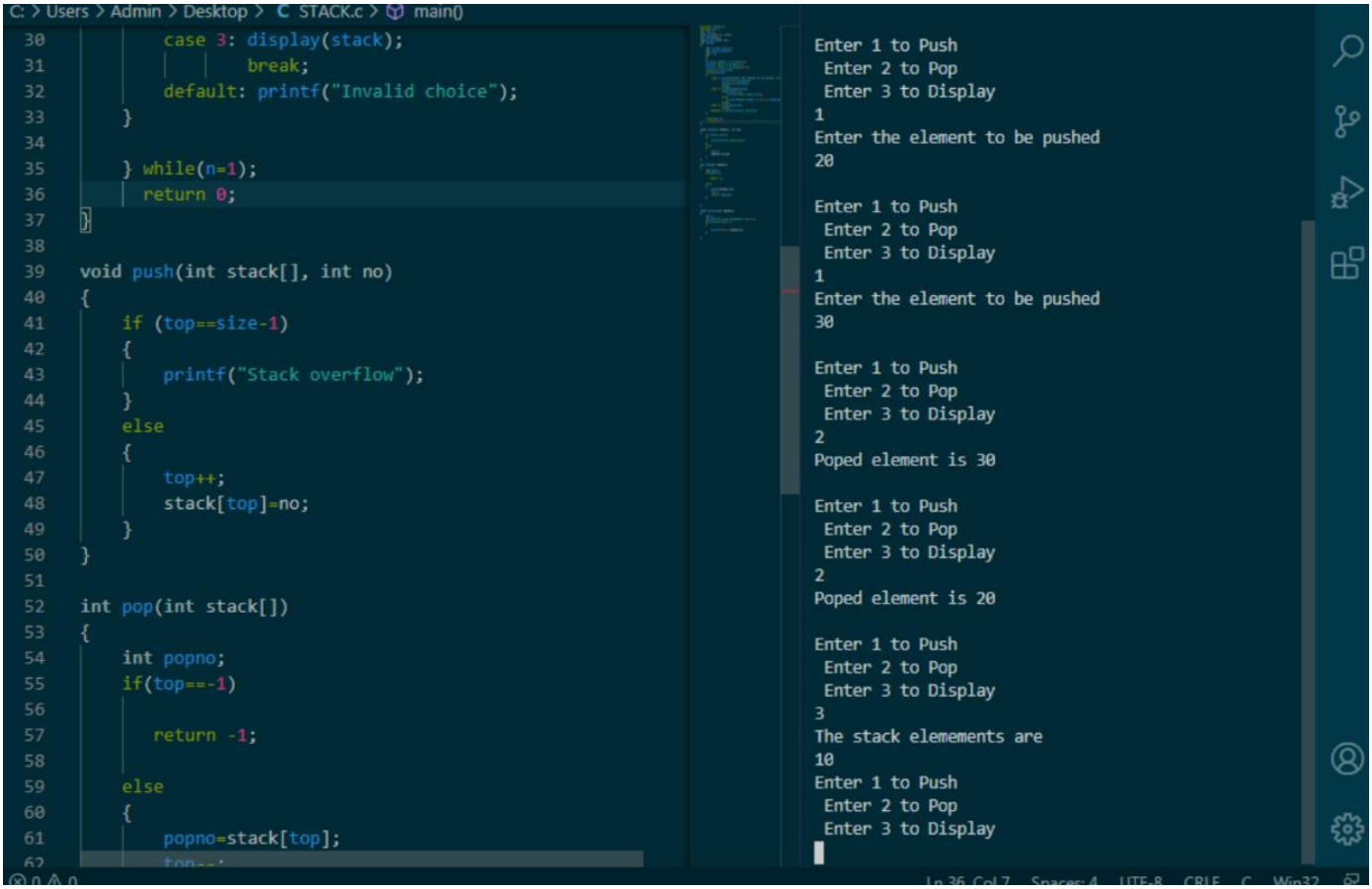
Final Marks Awarded

Obtained	Total

CONTENT

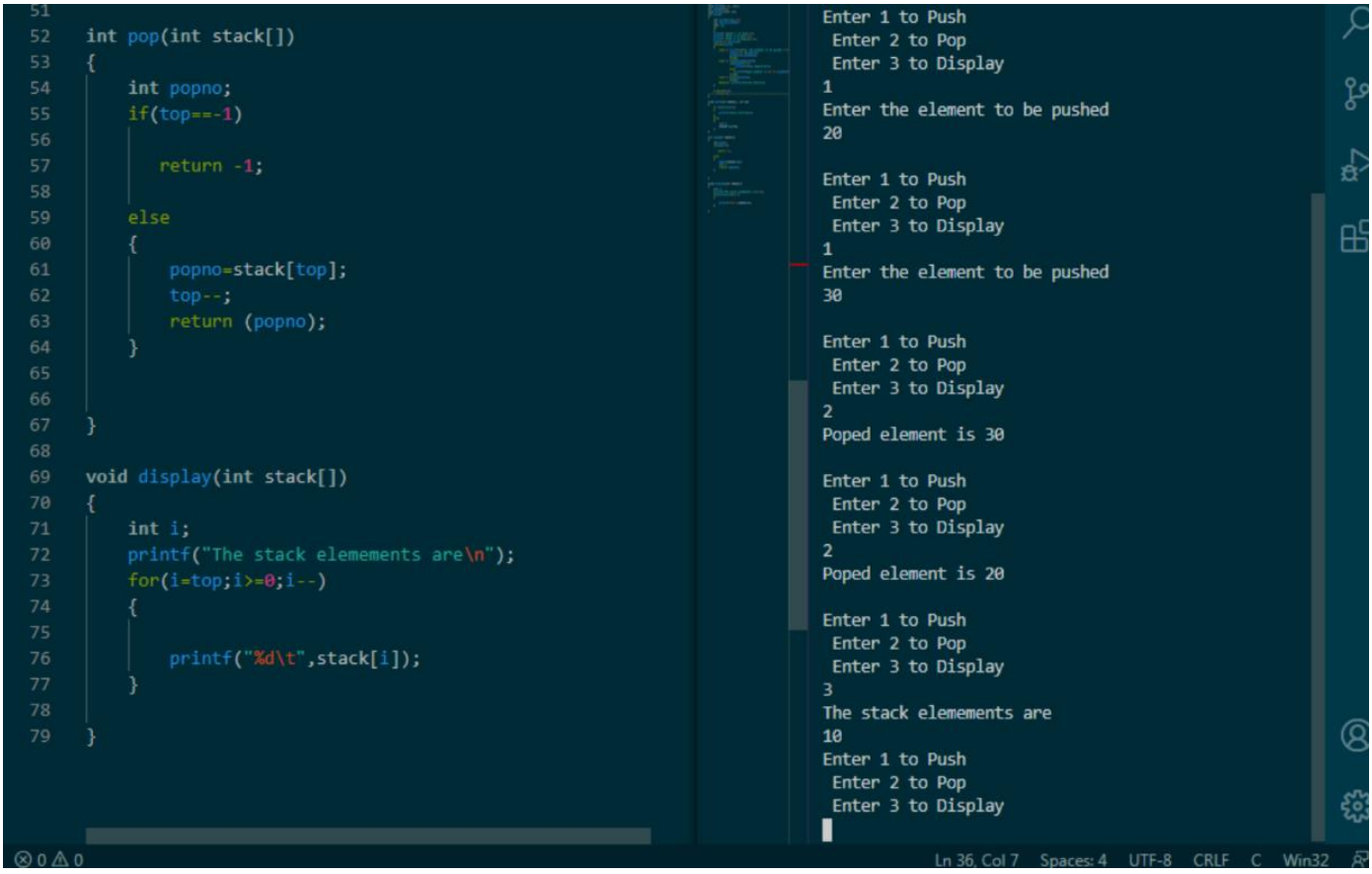
1. Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow





```
C: > Users > Admin > Desktop > C STACK.c > main()
30          case 3: display(stack);
31                  break;
32          default: printf("Invalid choice");
33      }
34
35      } while(n=1);
36      return 0;
37  }
38
39  void push(int stack[], int no)
40  {
41      if (top==size-1)
42      {
43          printf("Stack overflow");
44      }
45      else
46      {
47          top++;
48          stack[top]=no;
49      }
50  }
51
52  int pop(int stack[])
53  {
54      int popno;
55      if(top==-1)
56
57          return -1;
58
59      else
60      {
61          popno=stack[top];
62          top--;
```

```
Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
1
Enter the element to be pushed
20

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
1
Enter the element to be pushed
30

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
2
Poped element is 30

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
2
Poped element is 20

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
3
The stack elemements are
10
Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
```



```
52  int pop(int stack[])
53  {
54      int popno;
55      if(top==-1)
56
57          return -1;
58
59      else
60      {
61          popno=stack[top];
62          top--;
63          return (popno);
64      }
65
66
67  }
68
69  void display(int stack[])
70  {
71      int i;
72      printf("The stack elemements are\n");
73      for(i=top;i>=0;i--)
74      {
75
76          printf("%d\t",stack[i]);
77      }
78
79  }
```

```
Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
1
Enter the element to be pushed
20

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
1
Enter the element to be pushed
30

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
2
Poped element is 30

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
2
Poped element is 20

Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
3
The stack elemements are
10
Enter 1 to Push
 Enter 2 to Pop
 Enter 3 to Display
```

2.WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

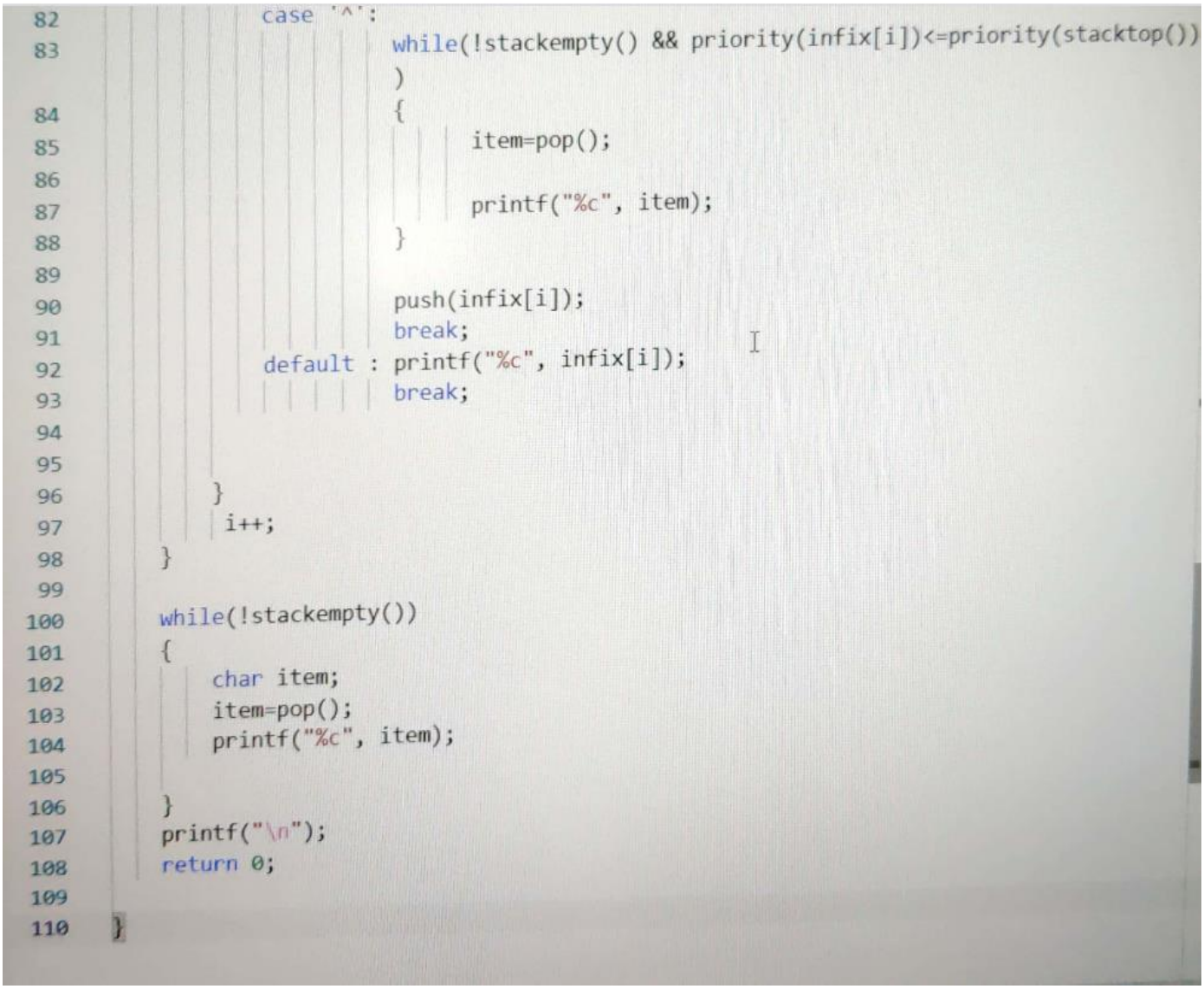
```
# define MAX 100
char stack[MAX];
int top=-1;

void push(char ch)
{
    if (top==MAX-1)
        printf("Stack is full\n");
    else
    {
        top++;
        stack[top]=ch;
    }
}
char pop()
{
    char item;
    if (top==-1)
        printf("\n stack is empty !");
    else
    {
        item=stack[top];
        top--;
        return item;
    }

}

int stackempty()
{
    if(top==-1) return 1;
```

main.c

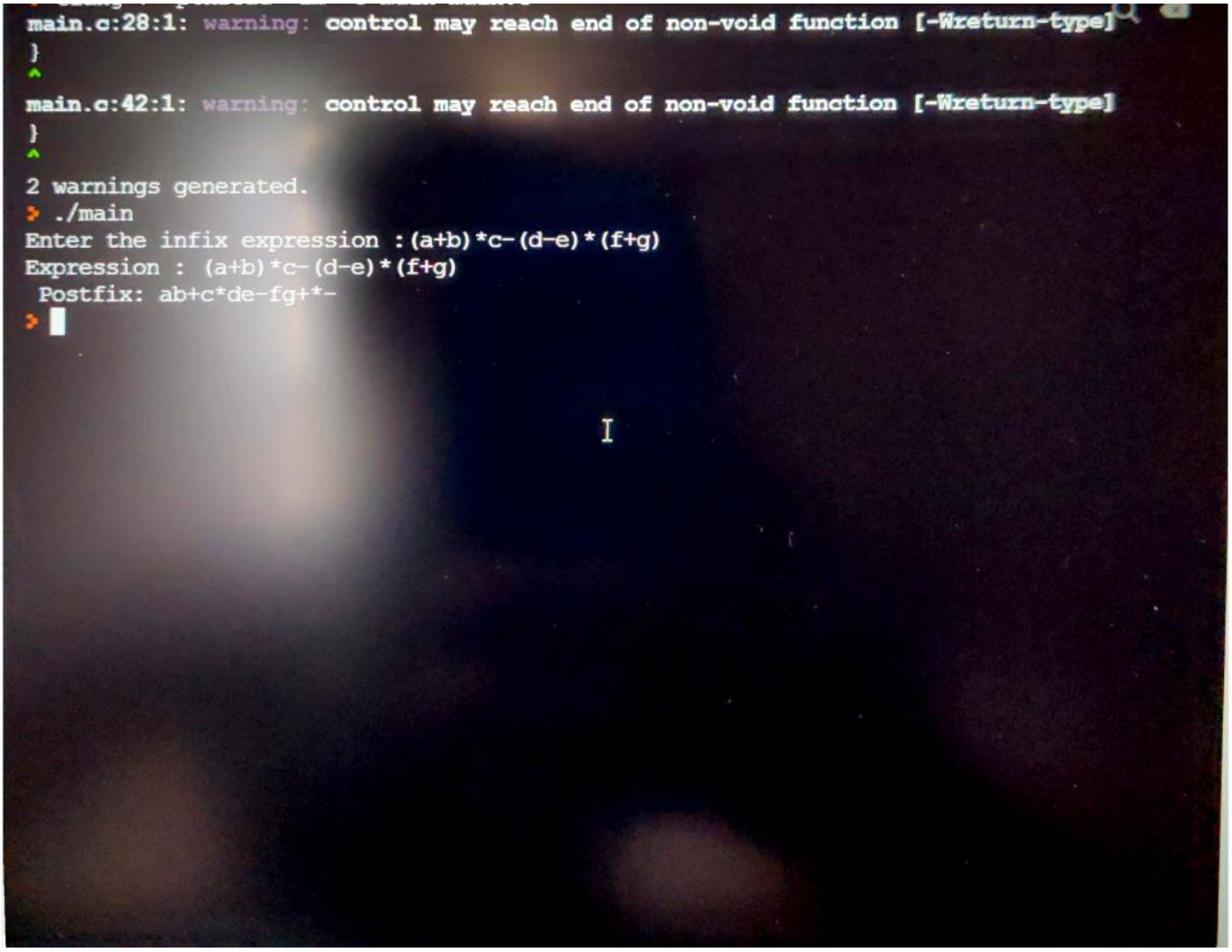
```
30  int stackempty()
31  {
32      if(top==-1) return 1;
33      else return 0;
34  }
35
36  char stacktop()
37  {
38      if( top==-1)
39          printf("\n stack is empty!");
40      else
41          return stack[top];
42  }
43  int priority(char ch)
44  {
45      switch(ch)
46      {
47          case '+':
48          case '-':return (1);
49          case '*':
50          case '/':return (2);
51          case '^': return (3);
52          default : return (0);
53      }
54  }
55
56
57
58  int main(int argc, char **argv)
59  {
```

main.c

```
char infix[100];
  int i, item;
  printf("Enter the infix expression :");
  scanf("%s",infix);
  printf("Expression : %s",infix);
  printf("\n Postfix: ");
  i=0;
   while (infix[i]!='\0')
  {


     switch (infix[i])
     {
        case '(': push(infix[i]);
                  break;
        case ')':while(( item=pop())!='(')
                    printf("%c",item);
                  break;
        case '+':
        case '-':
        case '*':
        case '/':
        case '^':
                  while(!stackempty() && priority(infix[i])<=priority(stacktop())
                  )
                  {
                     item=pop();

                     printf("%c", item);
                  }

                  push(infix[i]);
```



```
82                  case '^':
83                          while(!stackempty() && priority(infix[i])<=priority(stacktop())
                            )
84                          {
85                              item=pop();
86
87                              printf("%c", item);
88                          }
89
90                          push(infix[i]);
91                          break;
92                  default : printf("%c", infix[i]);
93                          break;
94
95
96          }
97           i++;
98      }
99
100     while(!stackempty())
101     {
102         char item;
103         item=pop();
104         printf("%c", item);
105
106     }
107     printf("\n");
108     return 0;
109
110 }
```



3.WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

main.c

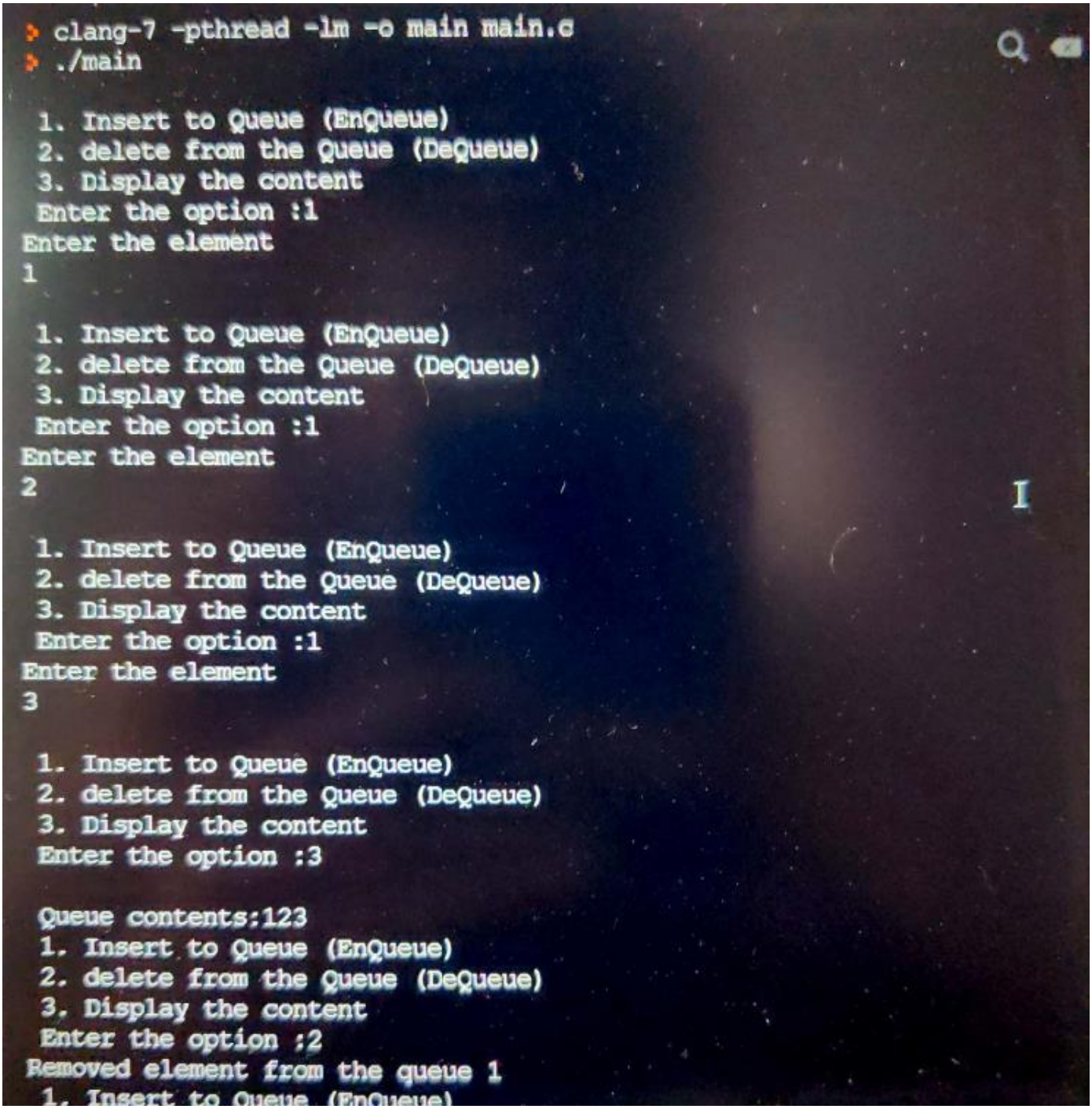
```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 5

int front=0;
int rear=-1;

int queue[MAX];

void Enque(int);
int Deque();
void display();
int main()
{
  int option;
    int item;
     int i=1;
    do{
        printf("\n 1. Insert to Queue (EnQueue)");
        printf("\n 2. delete from the Queue (DeQueue)");
        printf("\n 3. Display the content ");
        printf("\n Enter the option :");
        scanf("%d",&option);
        switch(option)
        {
            case 1:  printf("Enter the element\n");
                     scanf("%d",&item);
                     Enque(item);
                     break;
            case 2: item=Deque();
                    if(item==-1)
                        printf("Queue is empty\n");
```

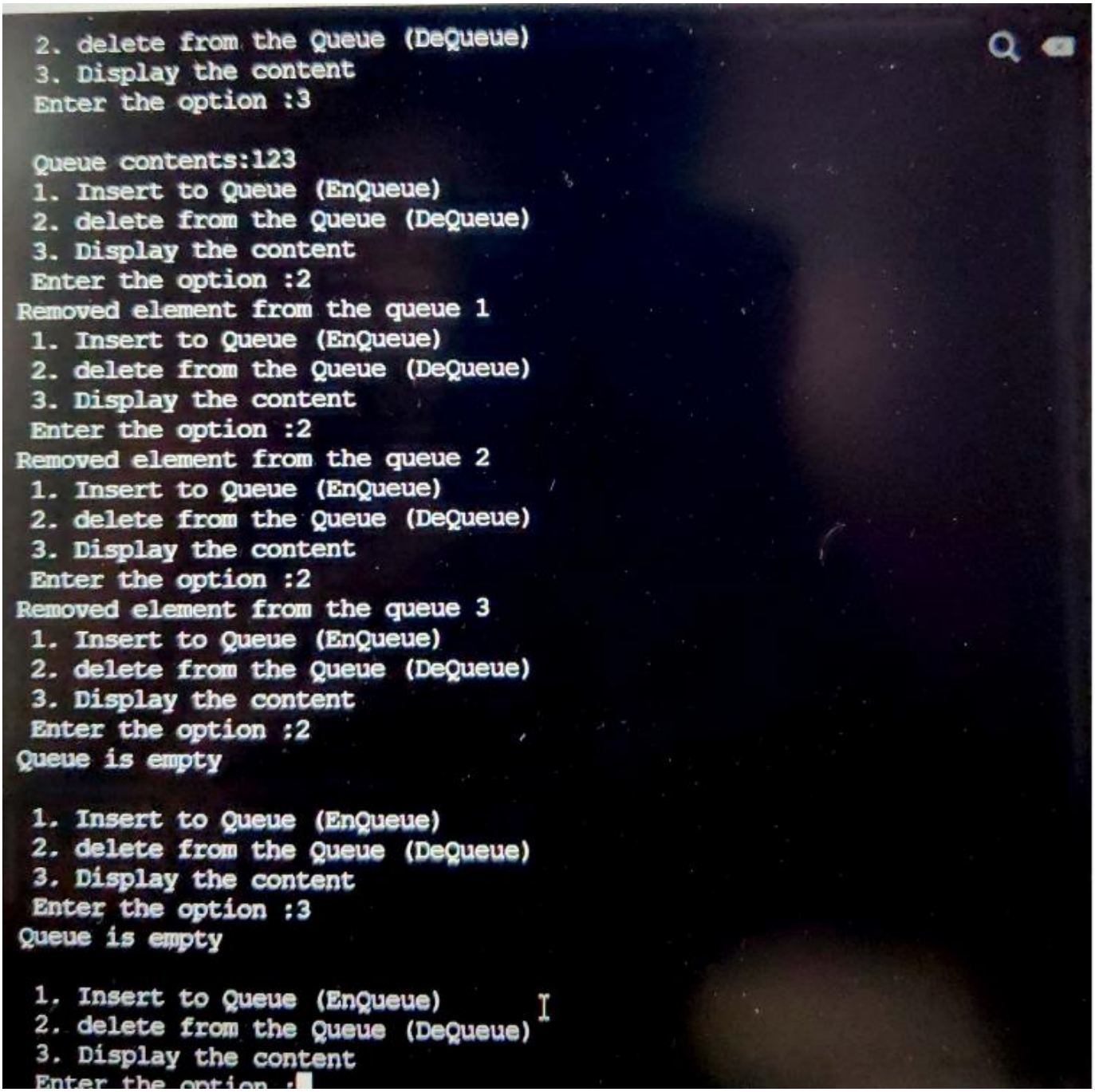
main.c

```
30                  case 2: item=Deque();
31                          if(item==-1)
32                              printf("Queue is empty\n");
33                          else
34                          printf("Removed element from the queue %d",item);
35                          break;
36                  case 3: display();
37                          break;
38              }
39      } while (i==1);
40    return 0;
41  }
42
43  void Enque(int ele)
44  {
45      if (rear==MAX-1)
46          printf("Queue is full\n");
47      else
48      {
49        rear++;
50        queue[rear]=ele;
51
52      }
53  }
54  int Deque()
55  {
56      int item;
57      if(front == -1)
58          return -1;
59      else
60      {
61          item=queue[front];
62          front++;
```

main.c

```
60      {
61          item=queue[front];
62          front++;
63          if(front>rear)
64          {
65              front=-1;
66              rear=-1;
67          }
68          return item;
69      }
70
71  }
72
73  void display()
74  {
75      int i;
76      if(front==-1)
77          printf("Queue is empty\n");
78      else
79      {
80          printf("\n Queue contents:");
81          for(i=front;i<=rear;i++)
82              printf("%d", queue[i]);
83      }
84  }
```


clang-7 -pthread -lm -o main main.c
./main
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
Enter the option :1
Enter the element
1
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
Enter the option :1
Enter the element
2
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
Enter the option :1
Enter the element
3
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
Enter the option :3
Queue contents:123
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
Enter the option :2
Removed element from the queue 1
1. Insert to Queue (EnQueue)



4.WAP to simulate the working of a circular queue of integers using an array. Provide the following operations. a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions

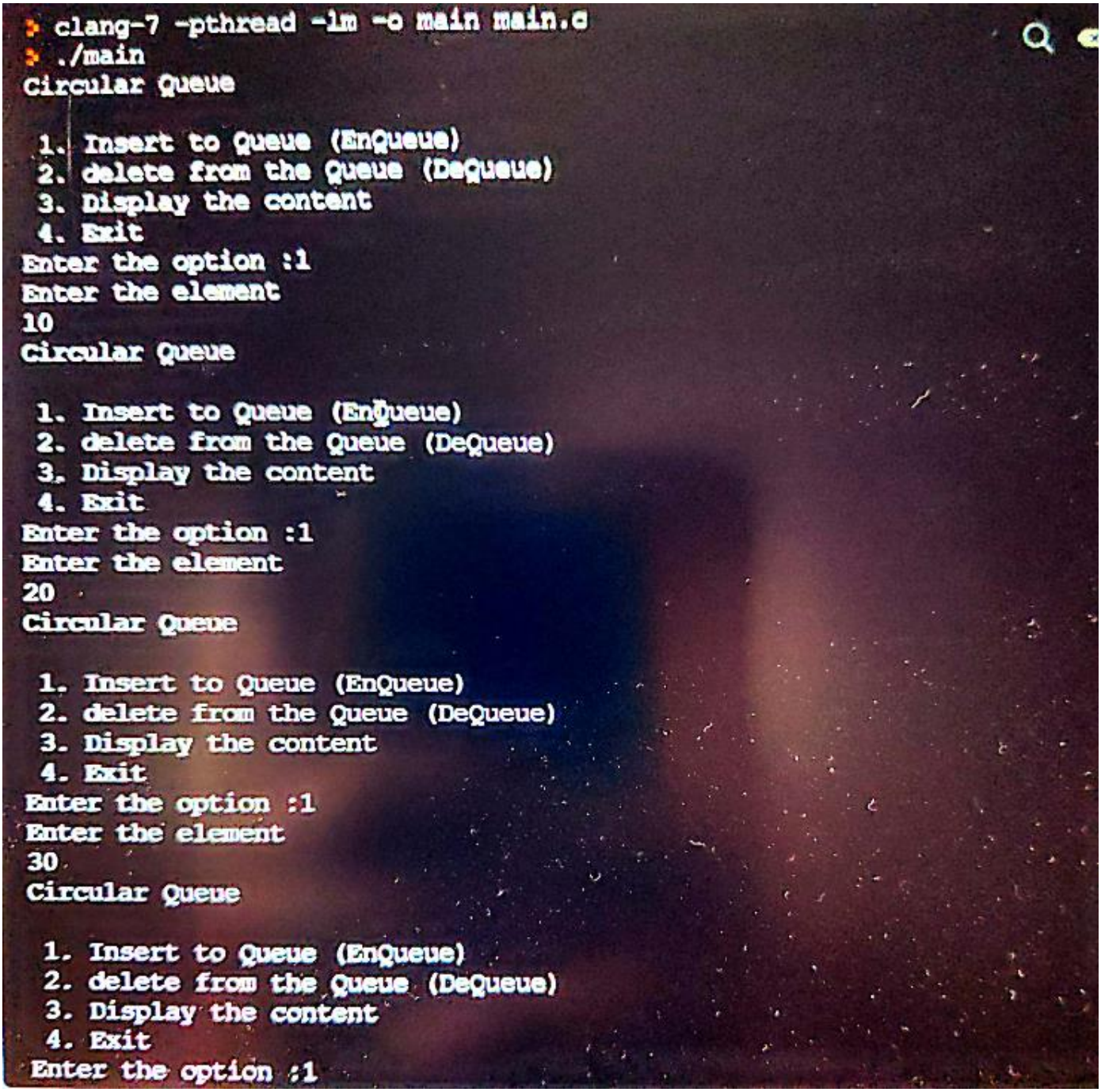
```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 3

int front=-1;
int rear=-1;

int queue[MAX];

void Enque(int);
int Deque();
void display();
int main(int argc, char **argv)
{
  int option;
    int item;
    do{
        printf("Circular Queue\n");
        printf("\n 1. Insert to Queue (EnQueue)");
        printf("\n 2. delete from the Queue (DeQueue)");
        printf("\n 3. Display the content ");
        printf("\n 4. Exit\n");
        printf("Enter the option :");
        scanf("%d",&option);
        switch(option)
        {
            case 1:  printf("Enter the element\n");
                     scanf("%d",&item);
                     Enque(item);
                     break;
            case 2: item=Deque();
                    if(item==-999)
```

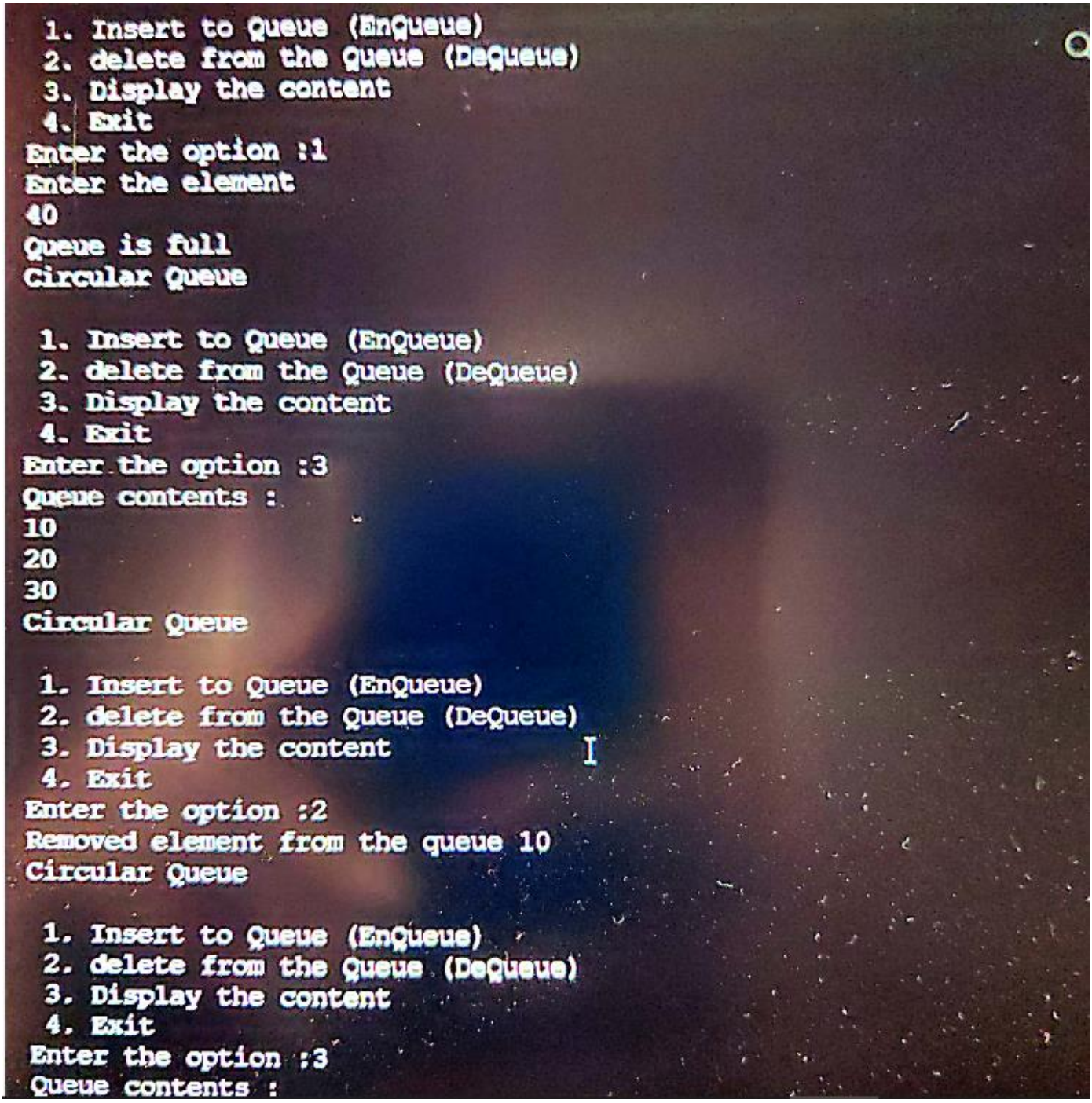
```
                break;
            case 2: item=Deque();
                if(item==-999)
                    printf("Queue is empty");
                else
                printf("Removed element from the queue %d",item);
                break;
            case 3: display();
                break;
            case 4: exit(0);
        }
    } while (option!=4);
  return 0;
}

void Enque(int ele)
{
    if(((front == 0 && rear == MAX - 1))|| (front == rear + 1) )
    {
        printf("Queue is full\n");return;

    }
    else
    {
      rear=(rear+1)%MAX;
      queue[rear]=ele;
      if(front ==-1)
          front=0;


    }
  }
  int Deque()
```

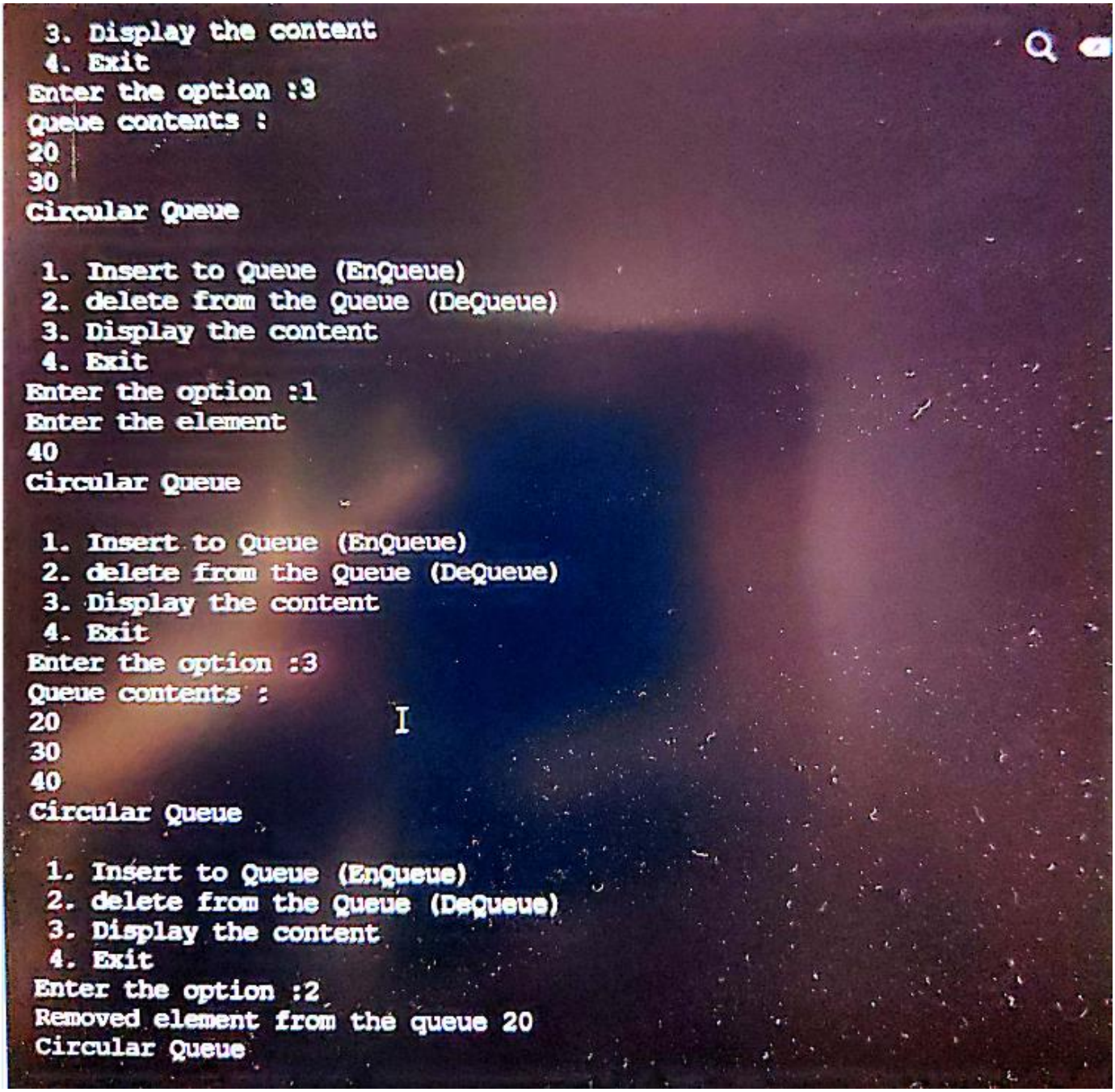
```
62  int Deque()
63  {
64      int item;
65      if((front == -1)&&(rear == -1))
66      {
67
68          return(-1);
69      }
70      else
71      {
72          item=queue[front];
73
74          if(front==rear)
75          {
76              front=-1;
77              rear=-1;
78          }
79          else
80          {
81              front=(front+1)%MAX;
82          }
83          return item;
84      }
85
86  }
87
88  void display()
89  {
90    if (front == -1 && rear == -1)
91    {
92      printf("Queue is empty");
93    }
```

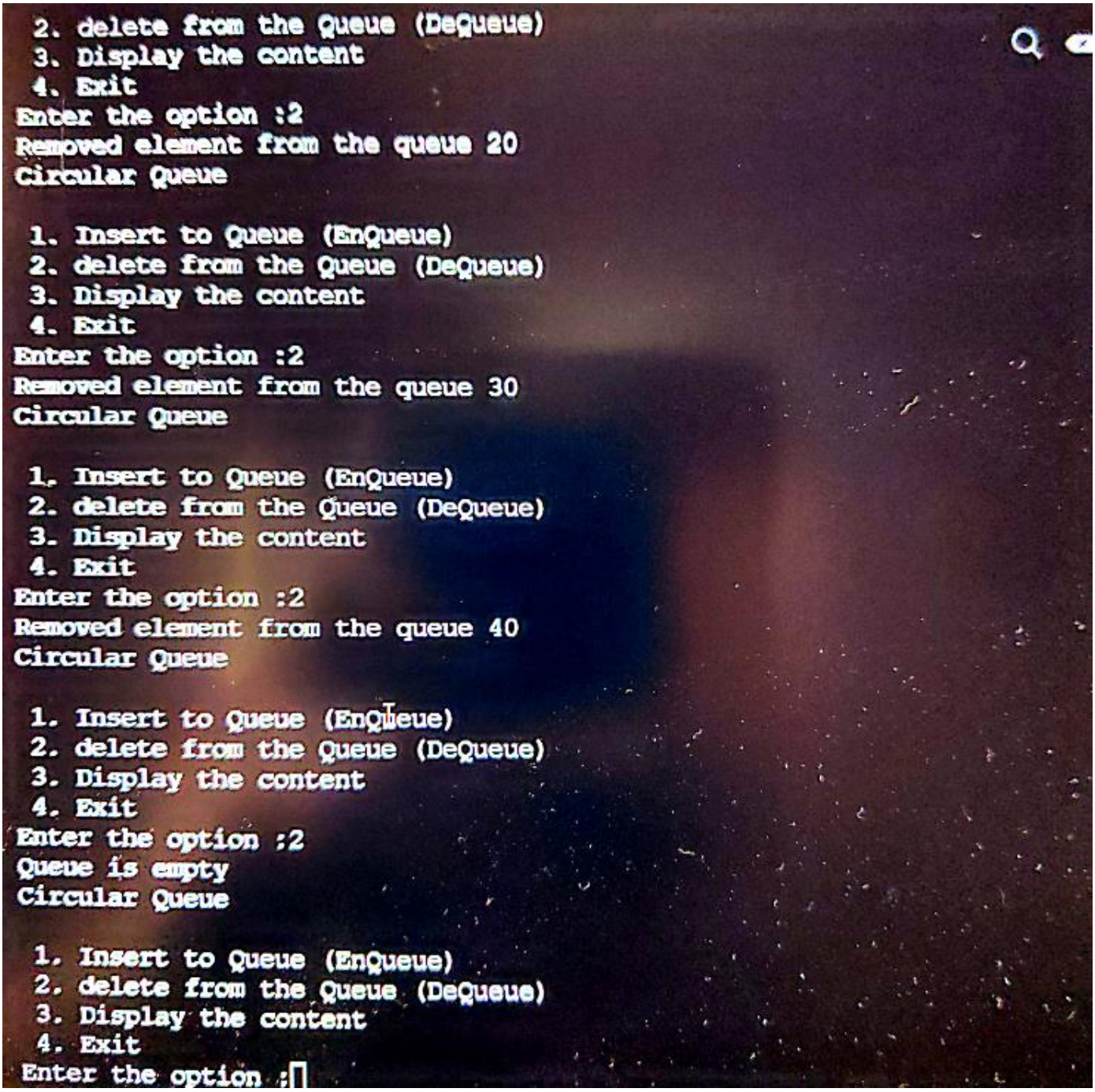
main.c

```
88  void display()
89  {
90      if (front == -1 && rear == -1)
91      {
92          printf("Queue is empty");
93      }
94      else
95      {
96          printf("Queue contents : \n");
97          if (front <= rear)
98          {
99              for (int i = front; i <= rear; i++)
100             {
101                 printf("%d\n", queue[i]);
102             }
103         }
104         else
105         {
106             for (int i = front; i <= MAX-1; i++)
107             {
108                 printf("%d\n", queue[i]);
109             }
110             for (int i = 0; i <= rear; i++)
111             {
112                 printf("%d\n", queue[i]);
113             }
114         }
115     }
116 }
117
```


clang-7 -pthread -lm -o main main.c
./main
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
10
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
20
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
30
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1


1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
40
Queue is full
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3
Queue contents :
10
20
30
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 10
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3
Queue contents :


3. Display the content
4. Exit
Enter the option :3
Queue contents :
20
30
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :1
Enter the element
40
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :3
Queue contents :
20
30
40
Circular Queue
1. Insert to Queue (EnQueue)
2. delete from the Queue (DeQueue)
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 20
Circular Queue



5.WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list.

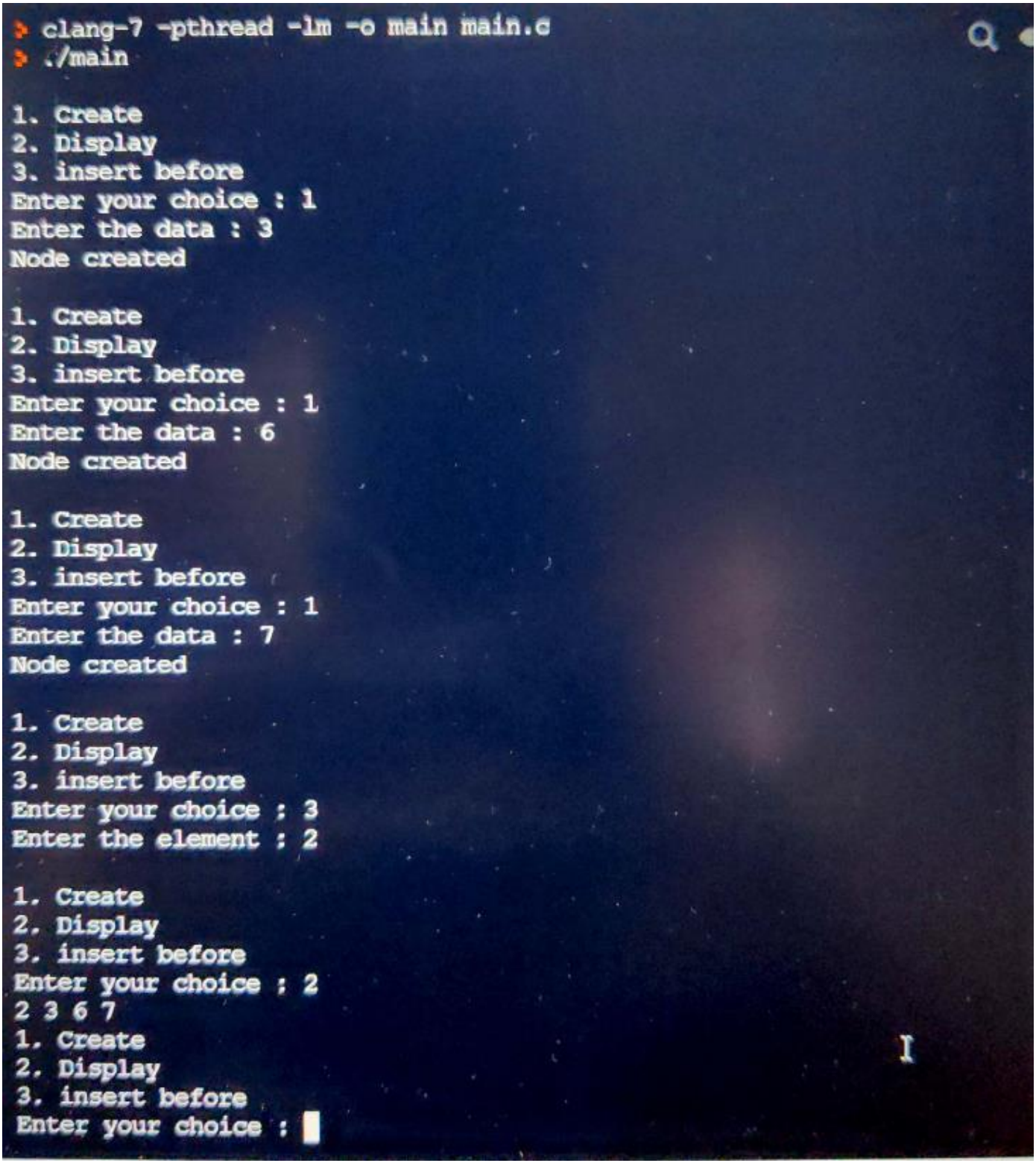
```c
#include <stdio.h>
#include <stdlib.h>
void create();
void display();
void insert_before();
struct node
{
    int data;
    struct node *next;
};
struct node *head=NULL;
int main()
{
    int choice,ele;
    char ch;
    int n=1;
    do
    {

    printf("\n1. Create \n2. Display  \n3. insert before ");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: create(); break;
        case 2: display();break;
        case 3: insert_before();
                break;
    }
    }while(n==1);
}
void create()
```

main.c

```
34      struct node *newnode,*temp;
35       int item;
36       newnode =(struct node *) malloc (sizeof(struct node));
37       printf("Enter the data : ");
38       scanf("%d",&item);
39       newnode->data=item;
40       if (head==NULL)
41        {
42         newnode->next=NULL;
43        head=newnode;
44        printf("Node created\n");
45        }
46        else
47       {
48       temp=head;
49          while(temp->next!=NULL)
50          {
51                    temp=temp->next;
52           }
53         temp->next=newnode;
54         newnode->next=NULL;
55          printf("Node created\n");
56      }
57  }
58
59  void display()
60  {
61       struct node *ptr=NULL;
62       ptr=head;
63
64       if(ptr==NULL)
65       {
```

main.c

```
66          printf("Nothing to print\n");
67      }
68      else
69      {
70          while(ptr!=NULL)
71          {
72          printf("%d ",ptr->data);
73          ptr=ptr->next;
74          }
75      }
76
77  }
78
79  void insert_before()
80   {
81          struct node *newnode;
82      int ele;
83      printf("Enter the element : ");
84      scanf("%d",&ele);
85
86      newnode=(struct node*)malloc(sizeof(struct node));
87
88      newnode->data =ele;
89      newnode->next=head;
90      head=newnode;
91
92
93   }
```



6.WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

main.c

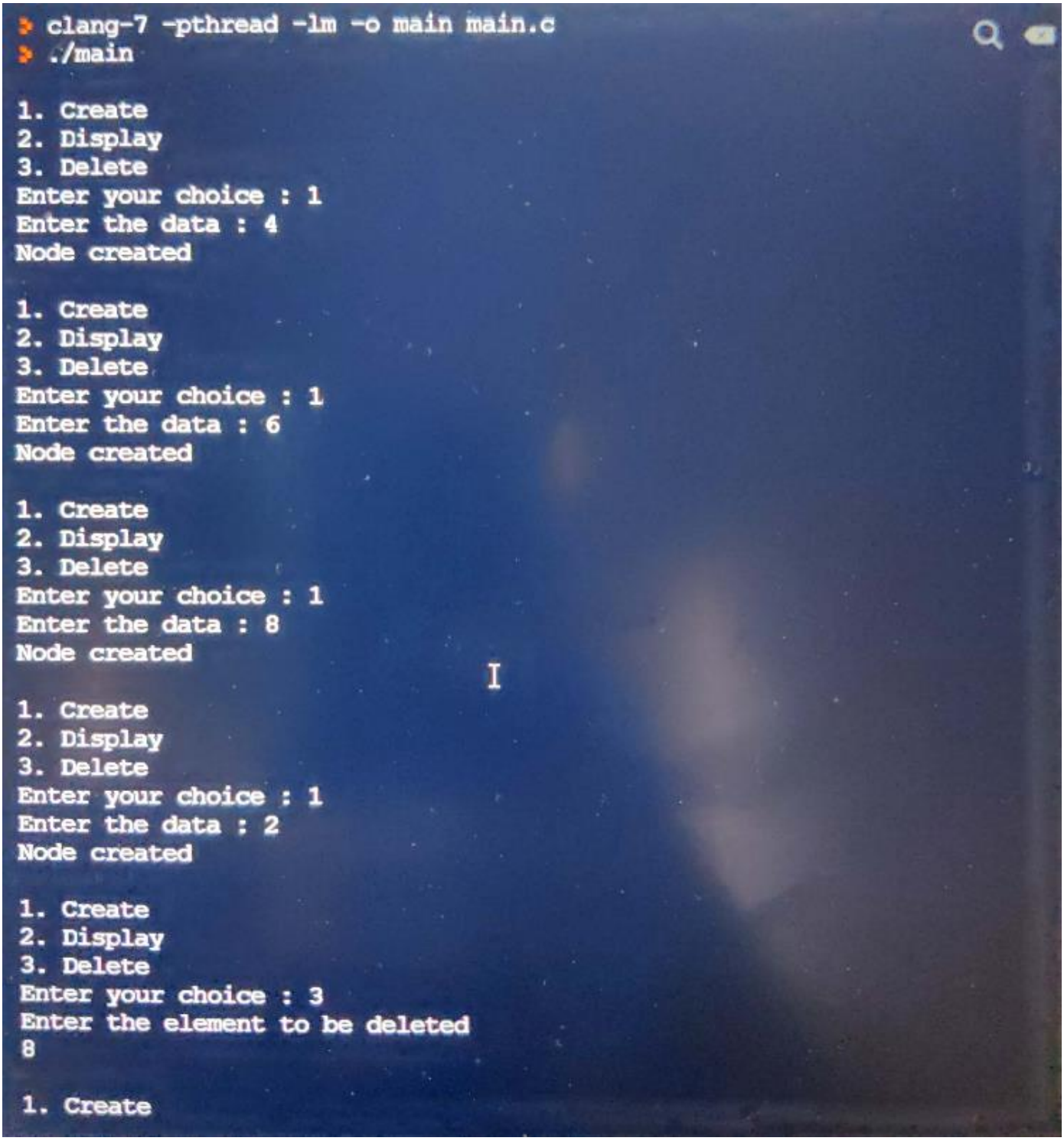
```c
#include <stdio.h>
#include <stdlib.h>
void create();
void display();
void delfun(int);
struct node
{
    int data;
    struct node *next;
};
struct node *head=NULL;
int main()
{
    int choice,ele;
    char ch;
    int n=1;
    do
    {

    printf("\n1. Create \n2. Display  \n3. Delete ");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: create(); break;
        case 2: display();break;
        case 3: printf("Enter the element to be deleted\n");
                scanf("%d",&ele);
                delfun(ele); break;

    }
    }while(n==1);
```

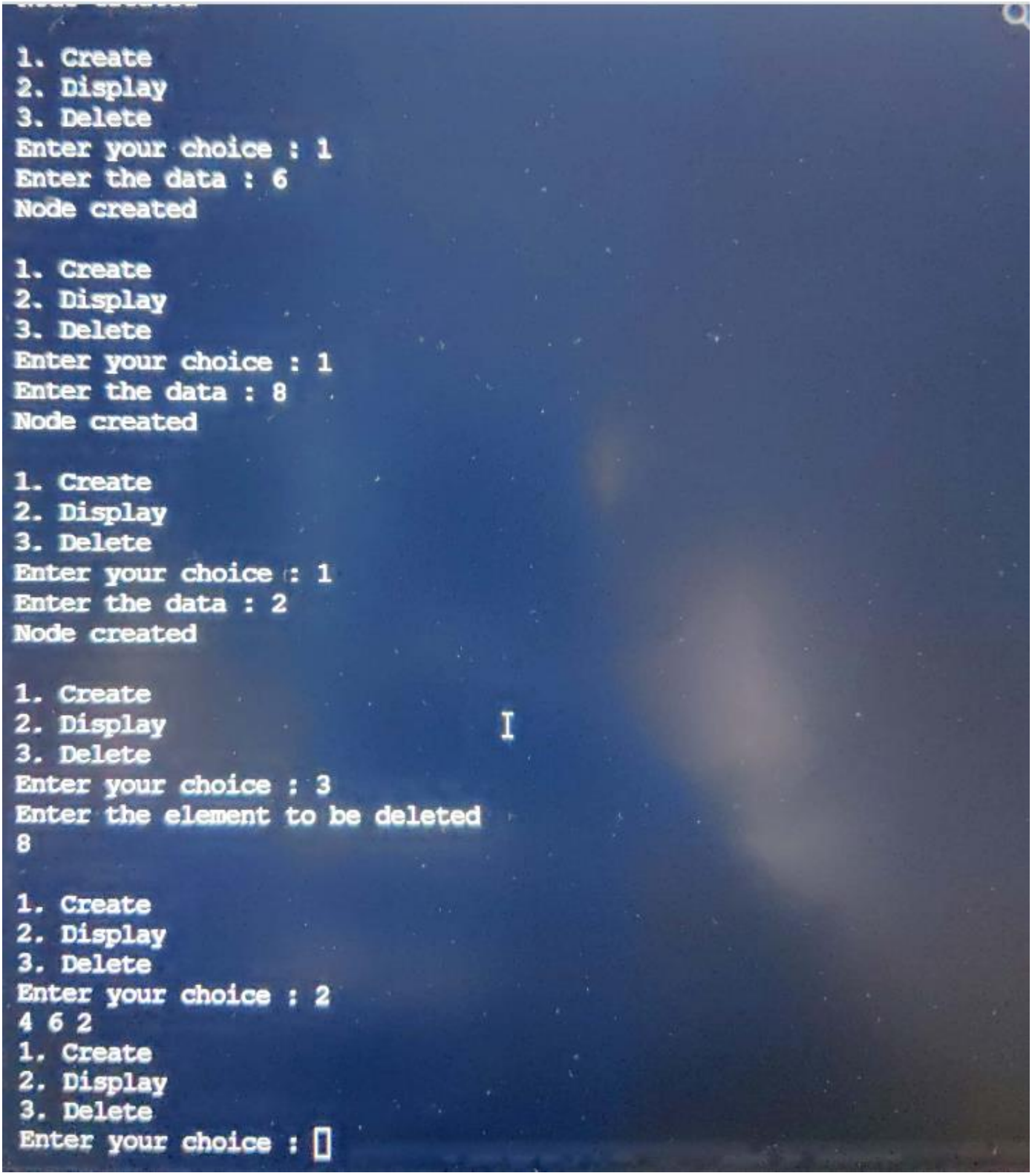
```
34  void create()
35  {
36      struct node *newnode,*temp;
37      int item;
38      newnode =(struct node *) malloc (sizeof(struct node));
39      printf("Enter the data : ");
40      scanf("%d",&item);
41      newnode->data=item;
42      if (head==NULL)
43      {
44          newnode->next=NULL;
45          head=newnode;
46          printf("Node created\n");
47      }
48      else
49      {
50      temp=head;
51          while(temp->next!=NULL)
52          {
53                  temp=temp->next;
54          }
55          temp->next=newnode;
56          newnode->next=NULL;
57          printf("Node created\n");
58      }
59  }
60
61  void display()
62  {
63      struct node *ptr=NULL;
64      ptr=head;
65
```

main.c

```
66      if(ptr==NULL)
67      {
68          printf("Nothing to print\n");
69      }
70      else
71      {
72          while(ptr!=NULL)
73         {
74          printf("%d ",ptr->data);
75          ptr=ptr->next;
76         }
77      }
78
79  }
80
81  void delfun(int ele)
82  {
83      struct node *temp,*del=NULL;
84
85       if (head == NULL)
86       {
87           printf("Empty List. Can't delete\n");return;
88       }
89       temp=head;
90       if(head->data==ele)
91       {
92           head=head->next;
93           return;
94       }
95      while (temp->next!=NULL)
96      {
97          if(temp->next->data==ele)
```

```
    if (head == NULL)
    {
        printf("Empty List. Can't delete\n");return;
    }
    temp=head;
    if(head->data==ele)
    {
        head=head->next;
        return;
    }
  while (temp->next!=NULL)
  {
      if(temp->next->data==ele)
      {
          del=temp->next;
         if(del->next==NULL)
         temp->next=NULL;
         else
          temp->next=del->next;
      }

      else
          temp=temp->next;
  }
  if(del==NULL)
  {
      printf("Element not found in the list\n");return;
}

}
```


clang-7 -pthread -lm -o main main.c
./main
1. Create
2. Display
3. Delete
Enter your choice : 1
Enter the data : 4
Node created
1. Create
2. Display
3. Delete
Enter your choice : 1
Enter the data : 6
Node created
1. Create
2. Display
3. Delete
Enter your choice : 1
Enter the data : 8
Node created
1. Create
2. Display
3. Delete
Enter your choice : 1
Enter the data : 2
Node created
1. Create
2. Display
3. Delete
Enter your choice : 3
Enter the element to be deleted
8
1. Create



7.WAP Implement Single Link List with following operations a) a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

main.c

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

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

void create(struct node **hptr);
void display(struct node *hptr);
void reverse(struct node **hptr);
void sort(struct node *hptr);
void concatenate(struct node *hptr1, struct node *hptr2);


int main(int argc, char **argv)
{
    struct node *head1=NULL;
    struct node *head2=NULL;
    int choice,ele,choice1;
    while(choice1!=4)
    {
        printf("1. List1 \n2. List2 \n3.Concatenate \n4.Display\n");
        printf("Enter your choice:");
        scanf("%d",&choice1);

        if(choice1 == 1)
        {
            printf("List1\n");
            while(choice!=5)
```

```
34          printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display
            \n5. Quit\n");
35          printf("Enter your choice : ");
36          scanf("%d",&choice);
37          if(choice == 1)
38          {
39              create(&head1);
40          }
41          else if(choice == 2)
42          {
43              sort(head1);
44          }
45          else if(choice == 3)
46          {
47              reverse(&head1);
48          }
49          else if(choice == 4)
50              display(head1);
51          else if(choice == 5)
52              break;
53          }
54      }
55      else if(choice1 == 2)
56      {
57          int choice2;
58          printf("List 2\n");
59        while(choice2!=5)
60          {
61          printf("\n1. Create \n2. Sort  \n3. Reverse  \n4. Display
            \n5. Quit");
62          printf("Enter your choice : ");
63          scanf("%d",&choice2);
```

main.c

```
64              if(choice2 == 1)
65              {
66                  create(&head2);
67              }
68              else if(choice2 == 2)
69              {
70                  sort(head2);
71              }
72              else if(choice2 == 3)
73              {
74                  reverse(&head2);
75              }
76              else if(choice2 == 4)
77                  display(head2);
78              else if(choice2 == 5)
79                  break;
80              }
81          }
82          else if(choice1 == 3)
83              concatenate(head1, head2);
84          else if(choice1 == 4)
85              display(head1);
86          else if(choice1 == 5)
87              break;
88      }
89      return 0;
90  }
91
92  void create(struct node **hptr)
93  {
94      struct node *newnode,*temp;
95       int item;
```

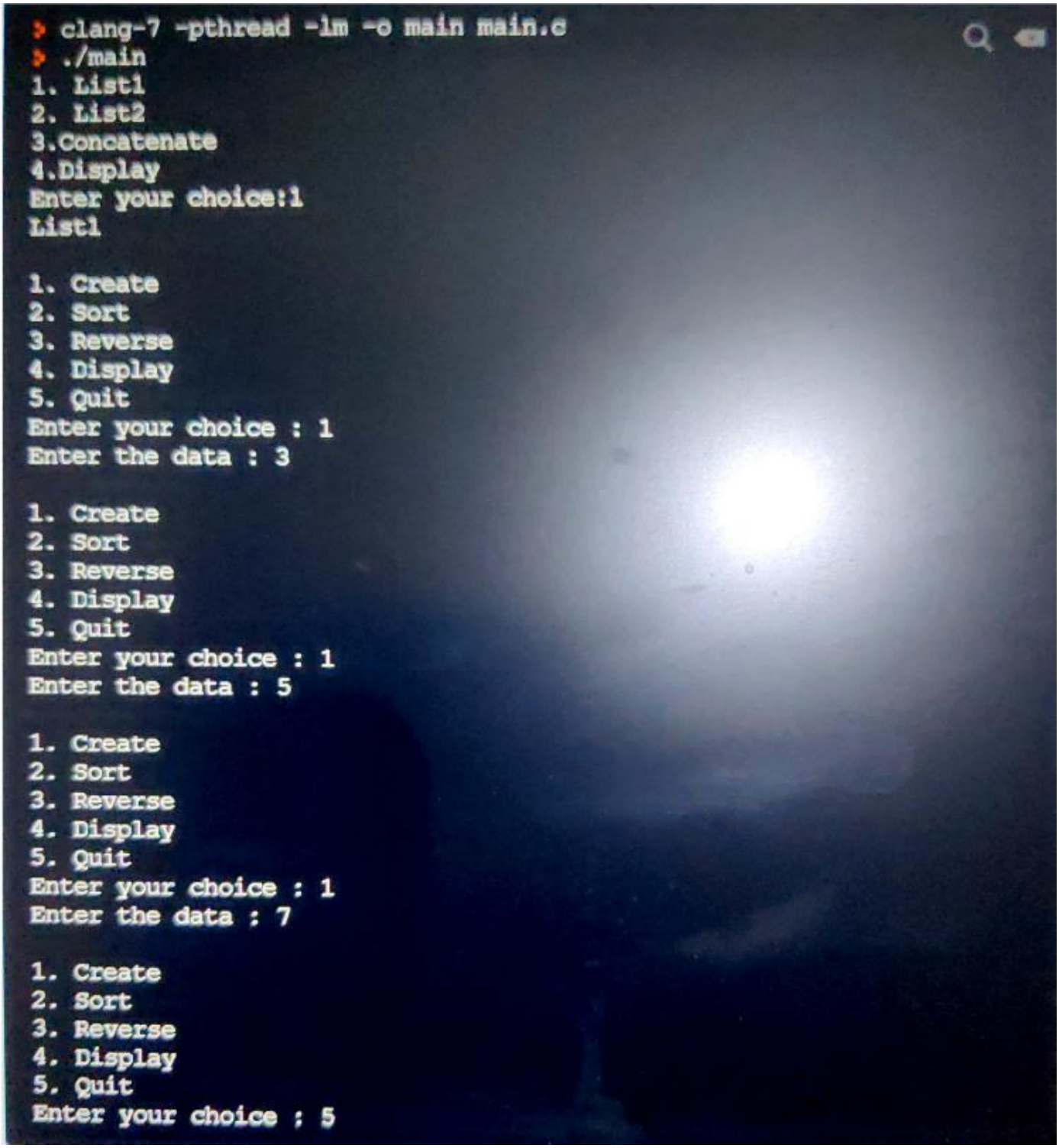
```
94      struct node *newnode,*temp;
95       int item;
96       newnode =(struct node *) malloc (sizeof(struct node));
97       printf("Enter the data : ");
98       scanf("%d",&item);
99       newnode->data=item;
100      if (*hptr==NULL)
101       {
102         newnode->next=NULL;
103        *hptr=newnode;
104       }
105      else
106      {
107      temp=*hptr;
108         while(temp->next!=NULL)
109         {
110                  temp=temp->next;
111          }
112        temp->next=newnode;
113        newnode->next=NULL;
114
115    }
116 }
117
118 void display(struct node *hptr)
119 {
120     struct node *ptr=NULL;
121     ptr=hptr;
122
123     if(ptr==NULL)
124     {
```

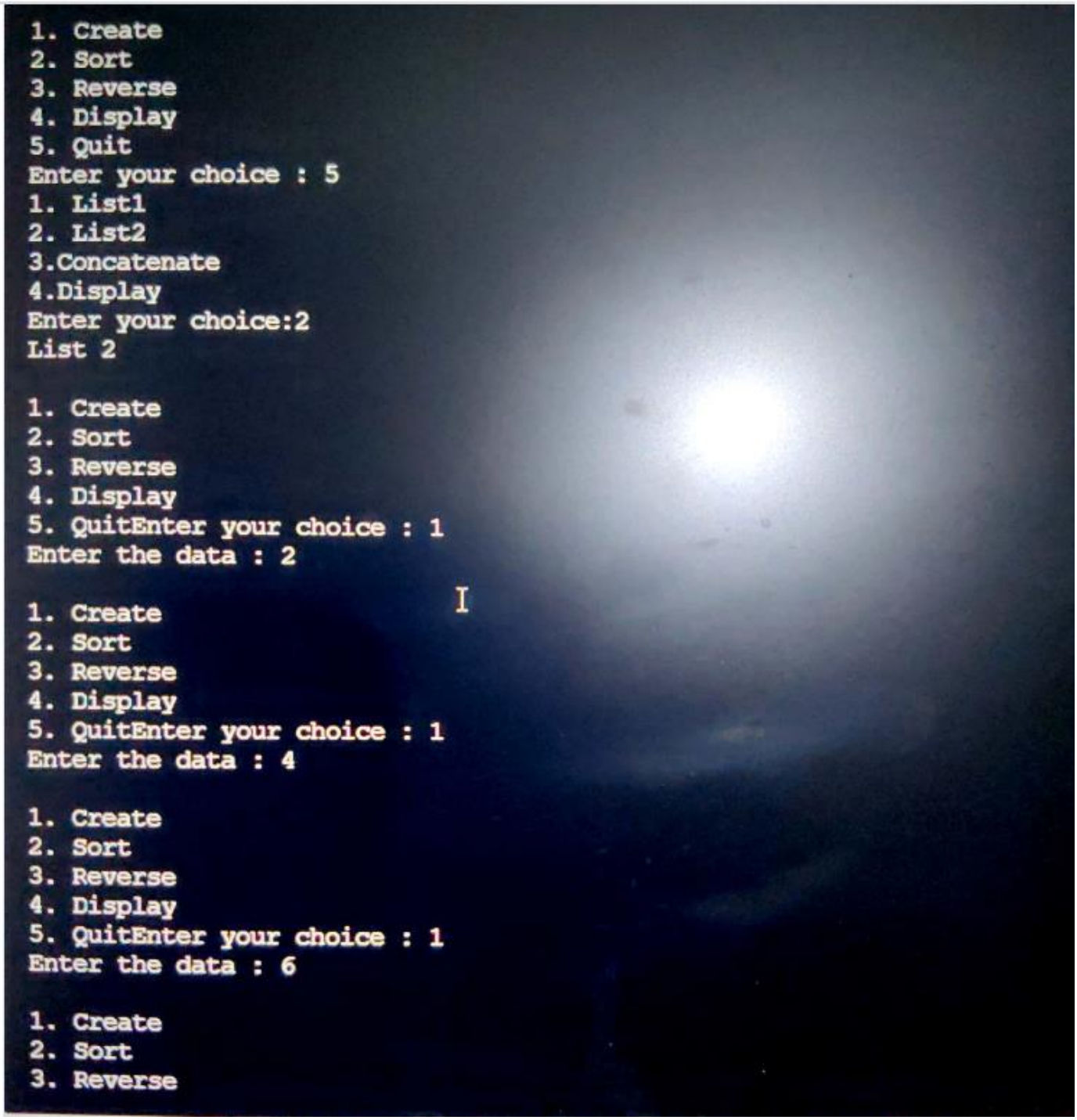
```
123         if(ptr==NULL)
124         {
125             printf("Nothing to print\n");
126         }
127         else
128         {
129             while(ptr!=NULL)
130             {
131             printf("%d ",ptr->data);
132             ptr=ptr->next;
133             }
134         }
135     }
136
137     void sort(struct node *hptr)
138     {
139         if(hptr == NULL)
140             printf("Empty list\n");
141         else
142         {
143             int swap;
144             struct node *first = NULL;
145             struct node *last = NULL;
146             do
147             {
148                 swap = 0;
149                 first = hptr;
150
151                 while(first->next != last)
152                 {
153                     if(first->data > first->next->data)
154                     {
```

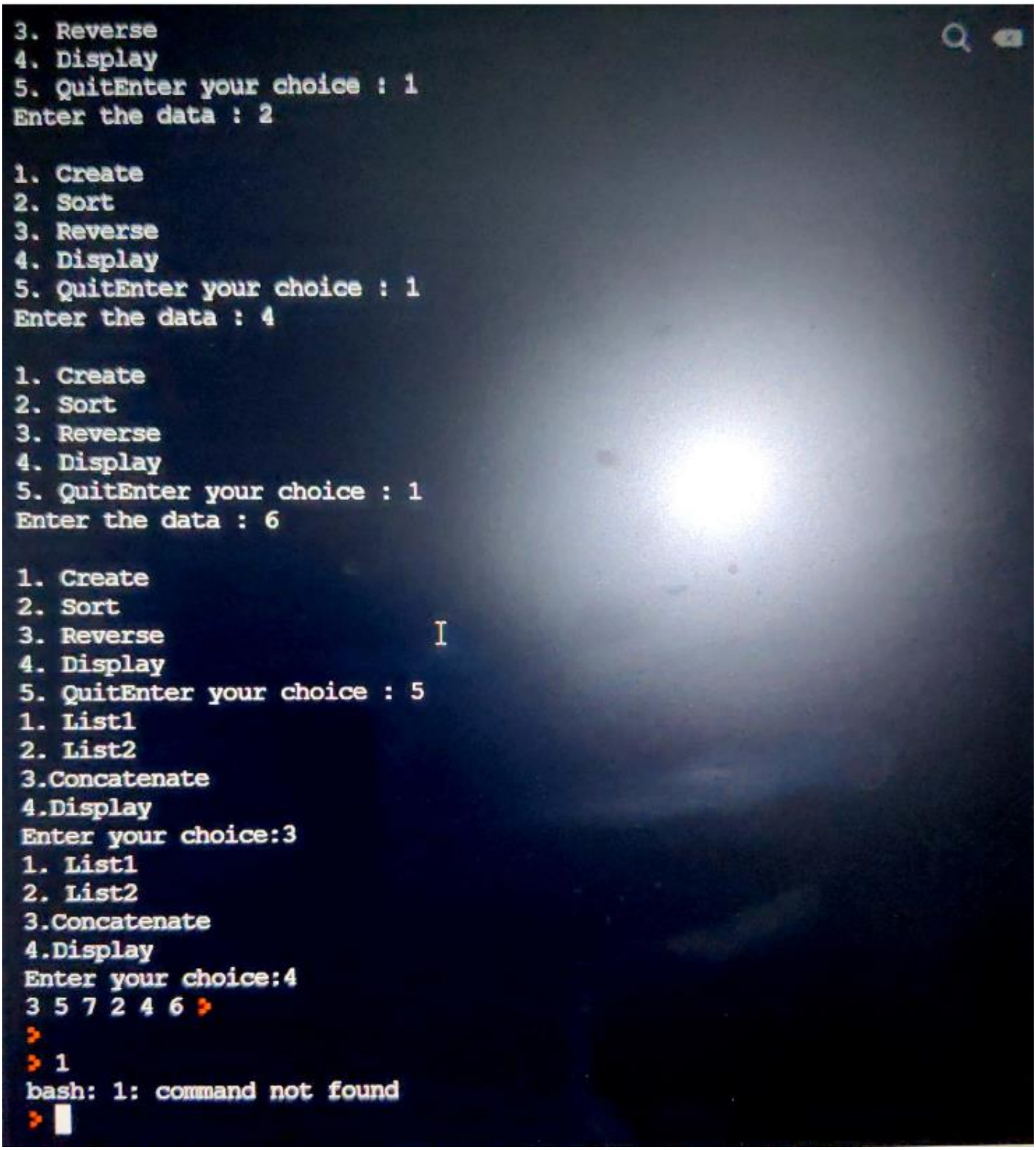
```
155                     int temp = first->data;
156                     first->data = first->next->data;
157                     first->next->data = temp;
158                     swap = 1;
159                 }
160                 first = first->next;
161             }
162             last = first;
163         }while(swap);
164     }
165 }
166
167 void reverse(struct node **hptr)
168 {
169     if(*hptr == NULL)
170     {
171         printf("Empty list\n");
172     }
173     else
174     {
175         struct node *prev, *curr, *head=*hptr;
176         prev = head;
177         curr = head->next;
178         head = head->next;
179
180         prev->next = NULL;
181
182         while(head!=NULL)
183         {
184             head = head->next;
185             curr->next = prev;
186
```

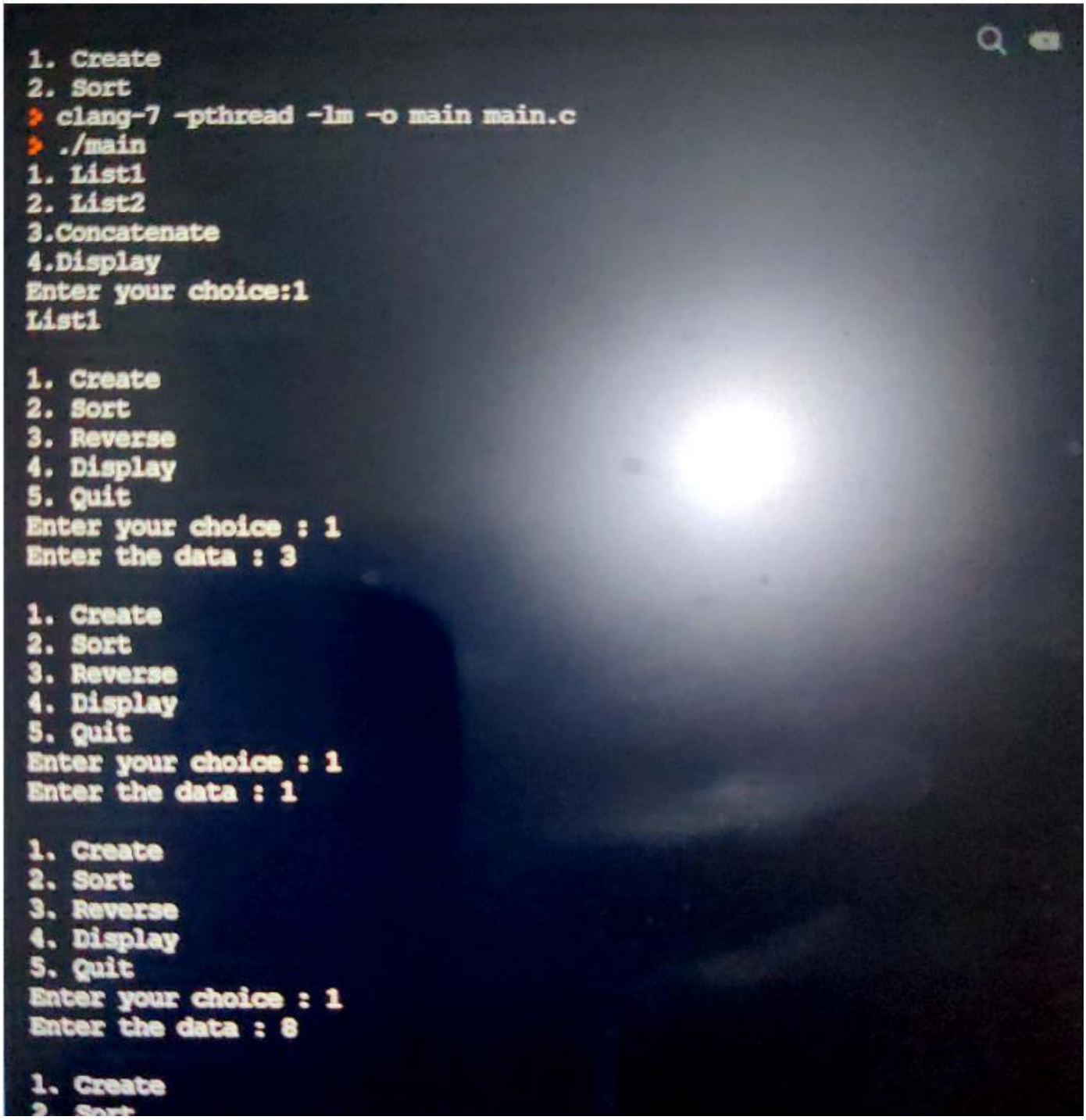
main.c

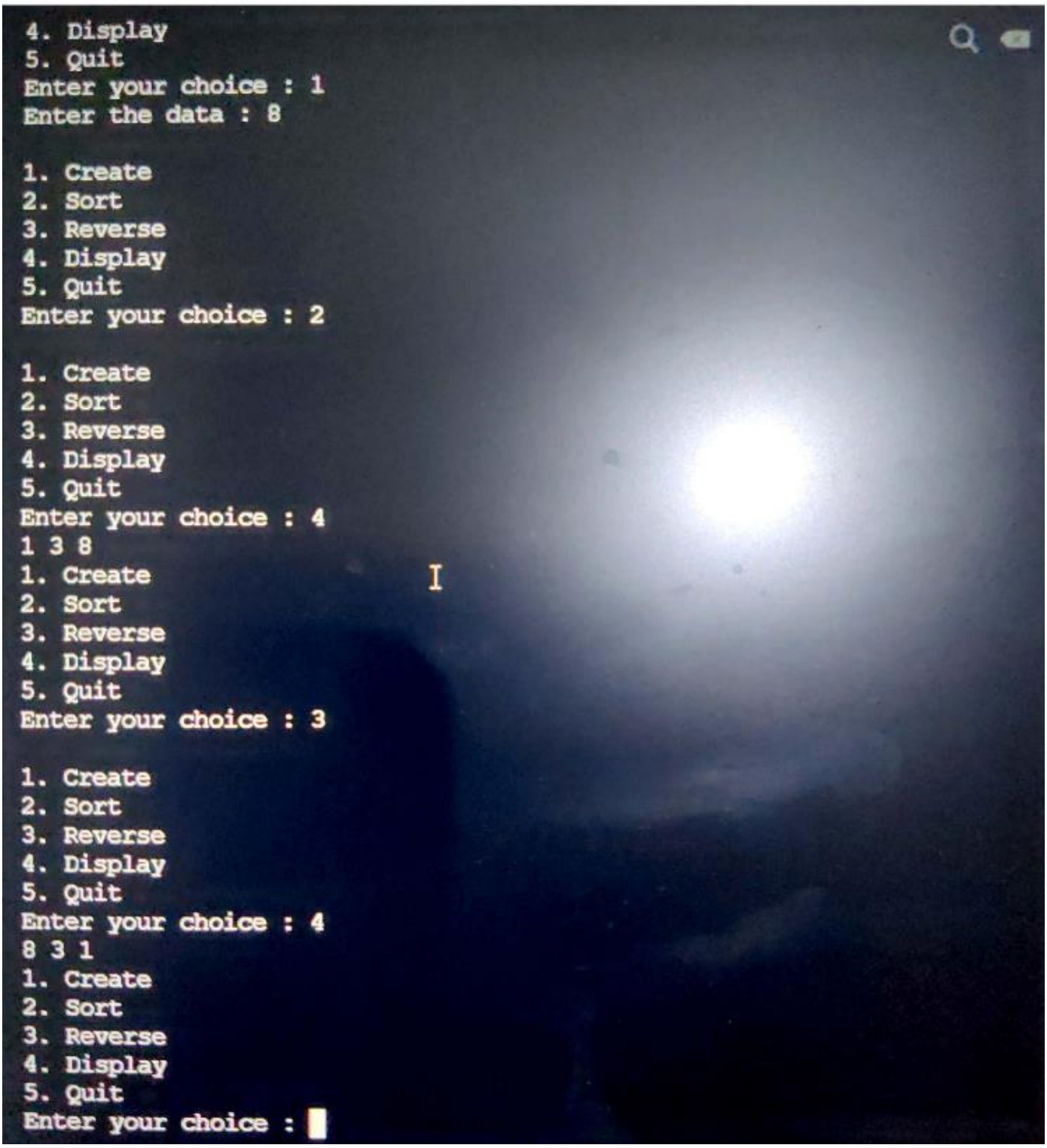
```
181
182             while(head!=NULL)
183             {
184                 head = head->next;
185                 curr->next = prev;
186
187                 prev = curr;
188                 curr = head;
189             }
190
191             *hptr = prev;
192         }
193     }
194
195     void concatenate(struct node *hptr1, struct node *hptr2)
196     {
197         if(hptr1 == NULL && hptr2 == NULL)
198             printf("Both are empty lists\n");
199         else if(hptr1 == NULL || hptr2 == NULL)
200             printf("One of them is empty\n");
201         else
202         {
203             struct node *temp1 = hptr1;
204             struct node *temp2 = hptr2;
205             while(temp1->next != NULL)
206                 temp1 = temp1->next;
207             temp1->next = temp2;
208         }
209     }
```


> clang-7 -pthread -lm -o main main.c
> ./main
1. List1
2. List2
3.Concatenate
4.Display
Enter your choice:1
List1
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 1
Enter the data : 3
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 1
Enter the data : 5
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 1
Enter the data : 7
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 5

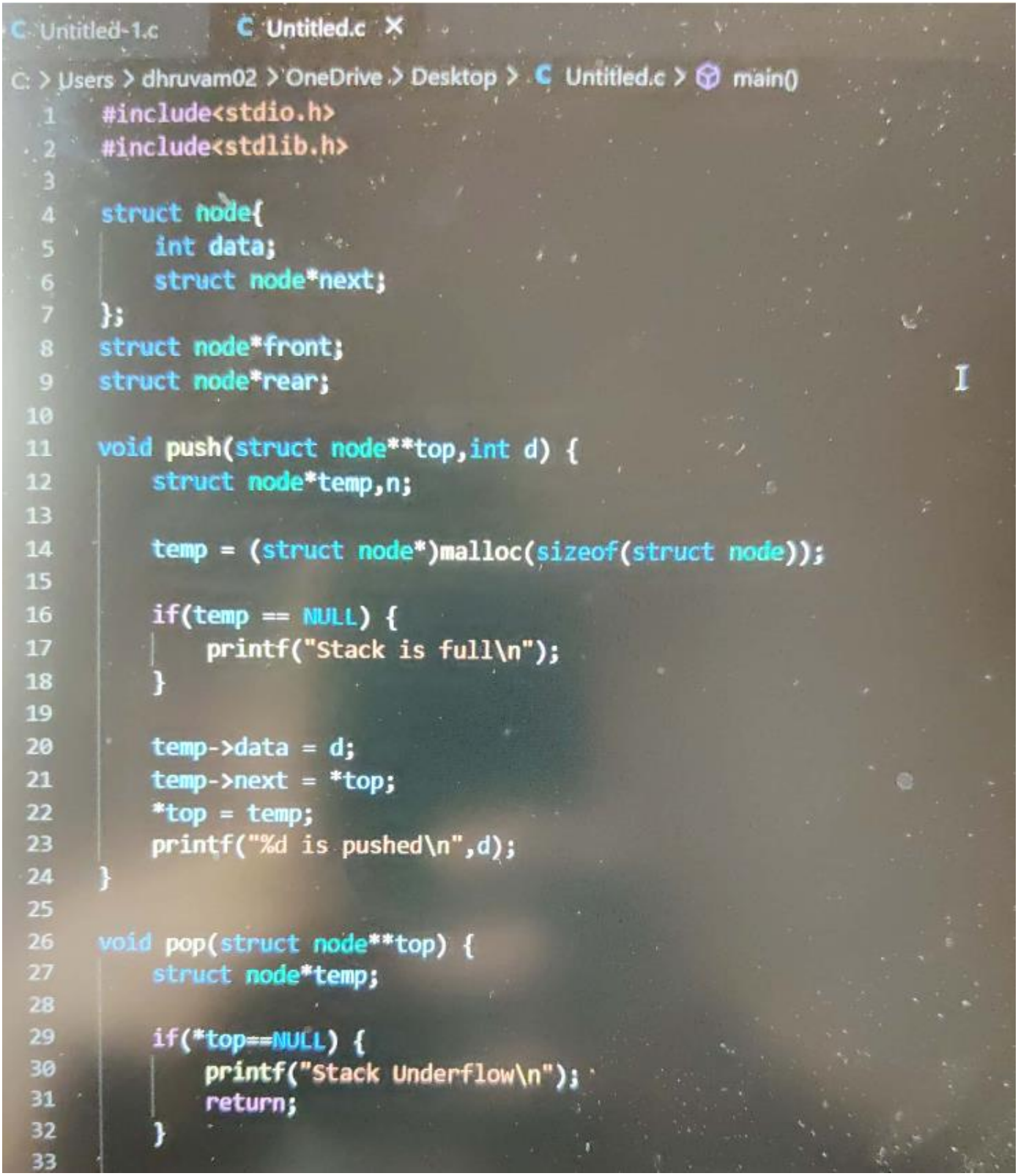

1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 5
1. List1
2. List2
3.Concatenate
4.Display
Enter your choice:2
List 2
1. Create
2. Sort
3. Reverse
4. Display
5. QuitEnter your choice : 1
Enter the data : 2
1. Create
2. Sort
3. Reverse
4. Display
5. QuitEnter your choice : 1
Enter the data : 4
1. Create
2. Sort
3. Reverse
4. Display
5. QuitEnter your choice : 1
Enter the data : 6
1. Create
2. Sort
3. Reverse

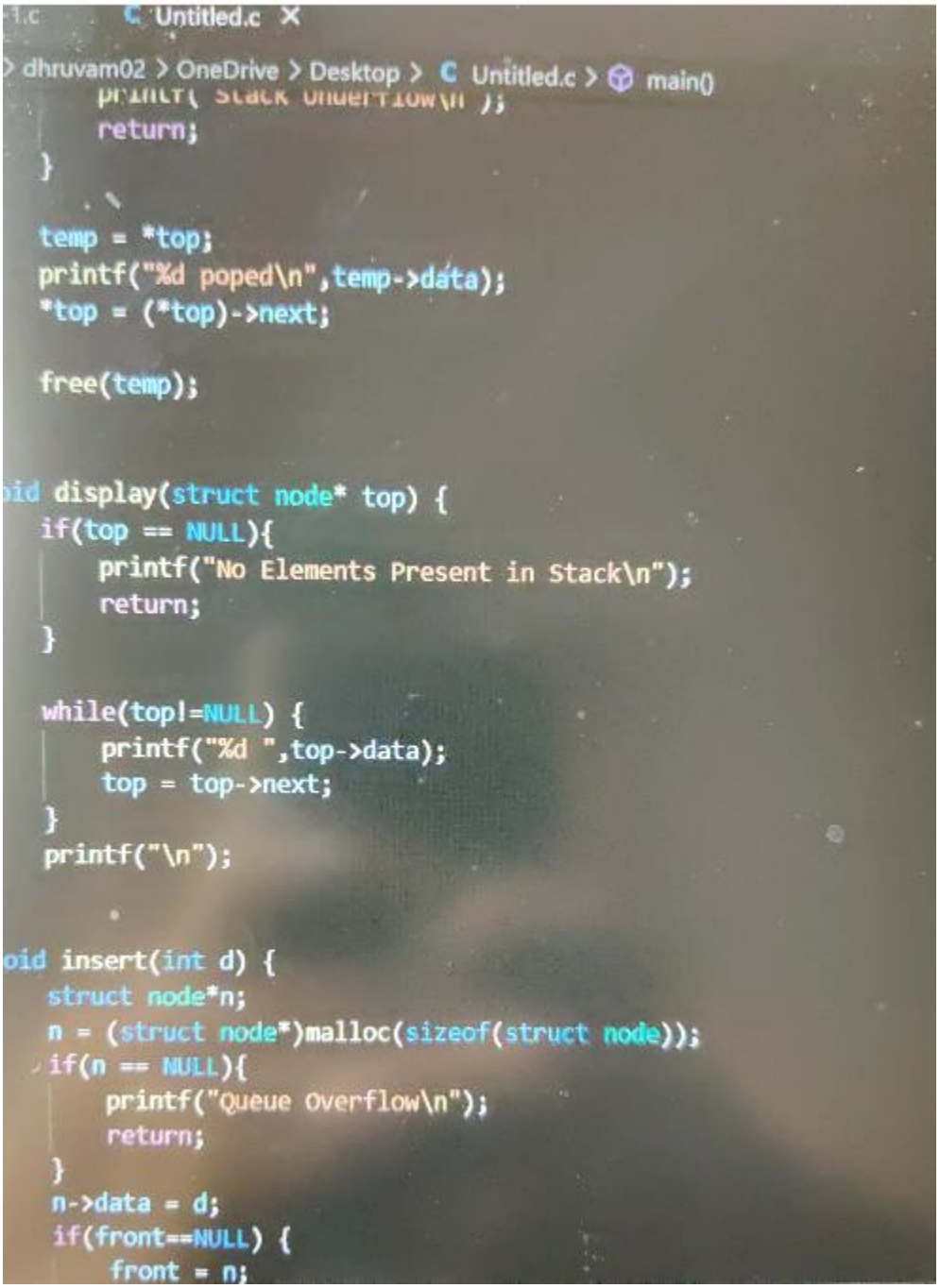

3. Reverse
4. Display
5. QuitEnter your choice : 1
Enter the data : 2
1. Create
2. Sort
3. Reverse
4. Display
5. QuitEnter your choice : 1
Enter the data : 4
1. Create
2. Sort
3. Reverse
4. Display
5. QuitEnter your choice : 1
Enter the data : 6
1. Create
2. Sort
3. Reverse
4. Display
5. QuitEnter your choice : 5
1. List1
2. List2
3.Concatenate
4.Display
Enter your choice:3
1. List1
2. List2
3.Concatenate
4.Display
Enter your choice:4
3 5 7 2 4 6
1
bash: 1: command not found

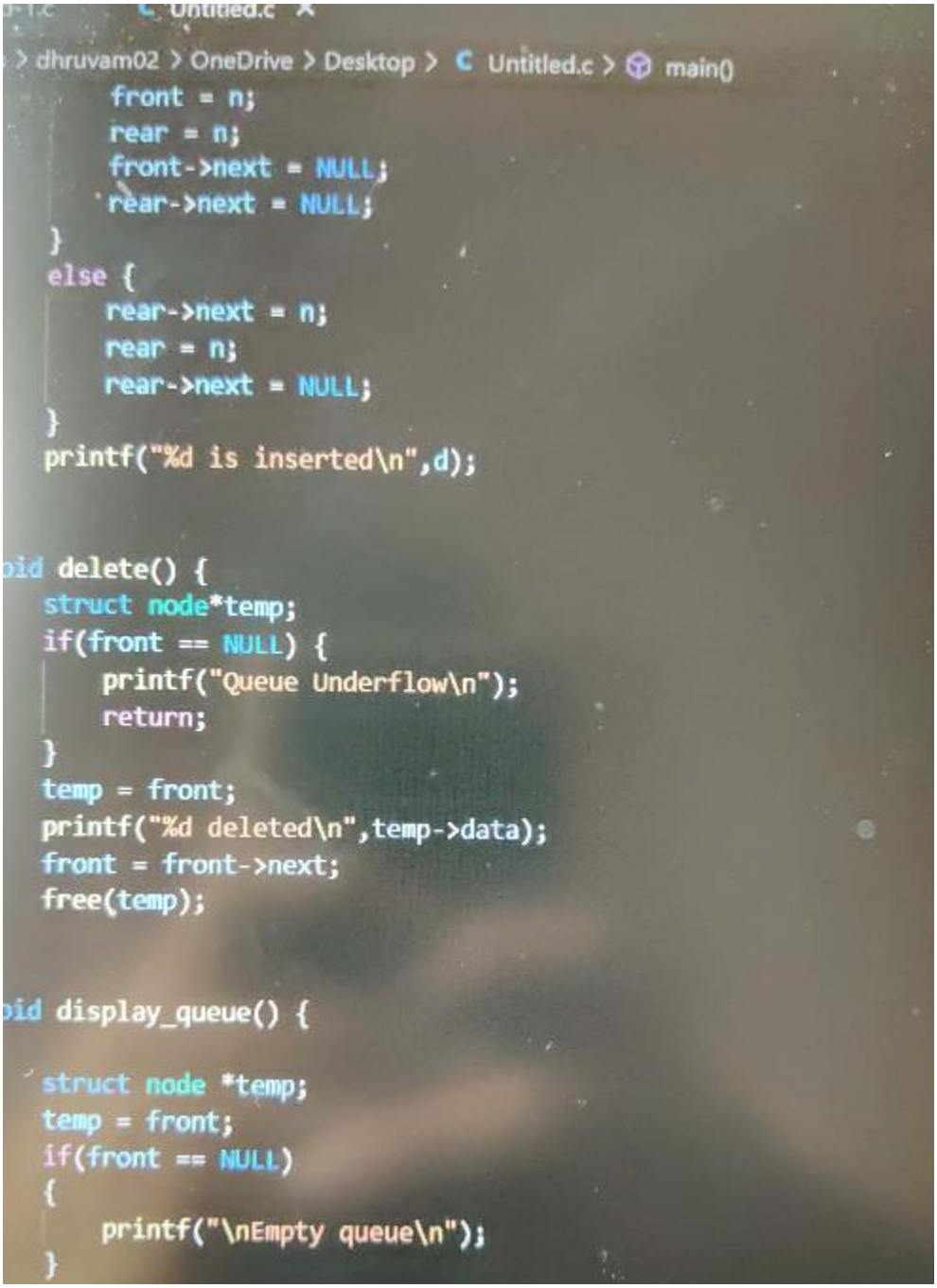

1. Create
2. Sort
> clang-7 -pthread -lm -o main main.c
> ./main
1. List1
2. List2
3.Concatenate
4.Display
Enter your choice:1
List1
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 1
Enter the data : 3
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 1
Enter the data : 1
1. Create
2. Sort
3. Reverse
4. Display
5. Quit
Enter your choice : 1
Enter the data : 8
1. Create

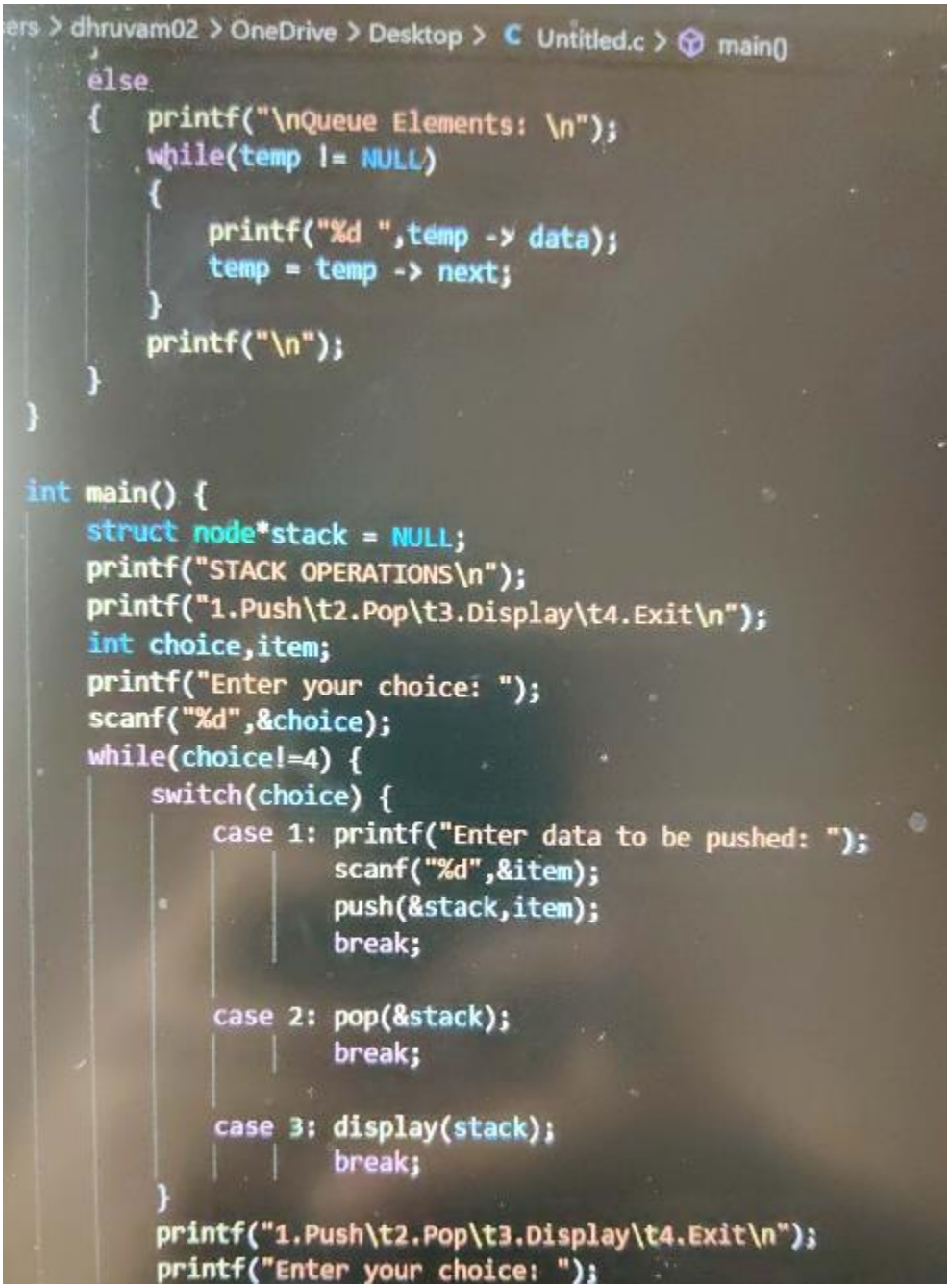


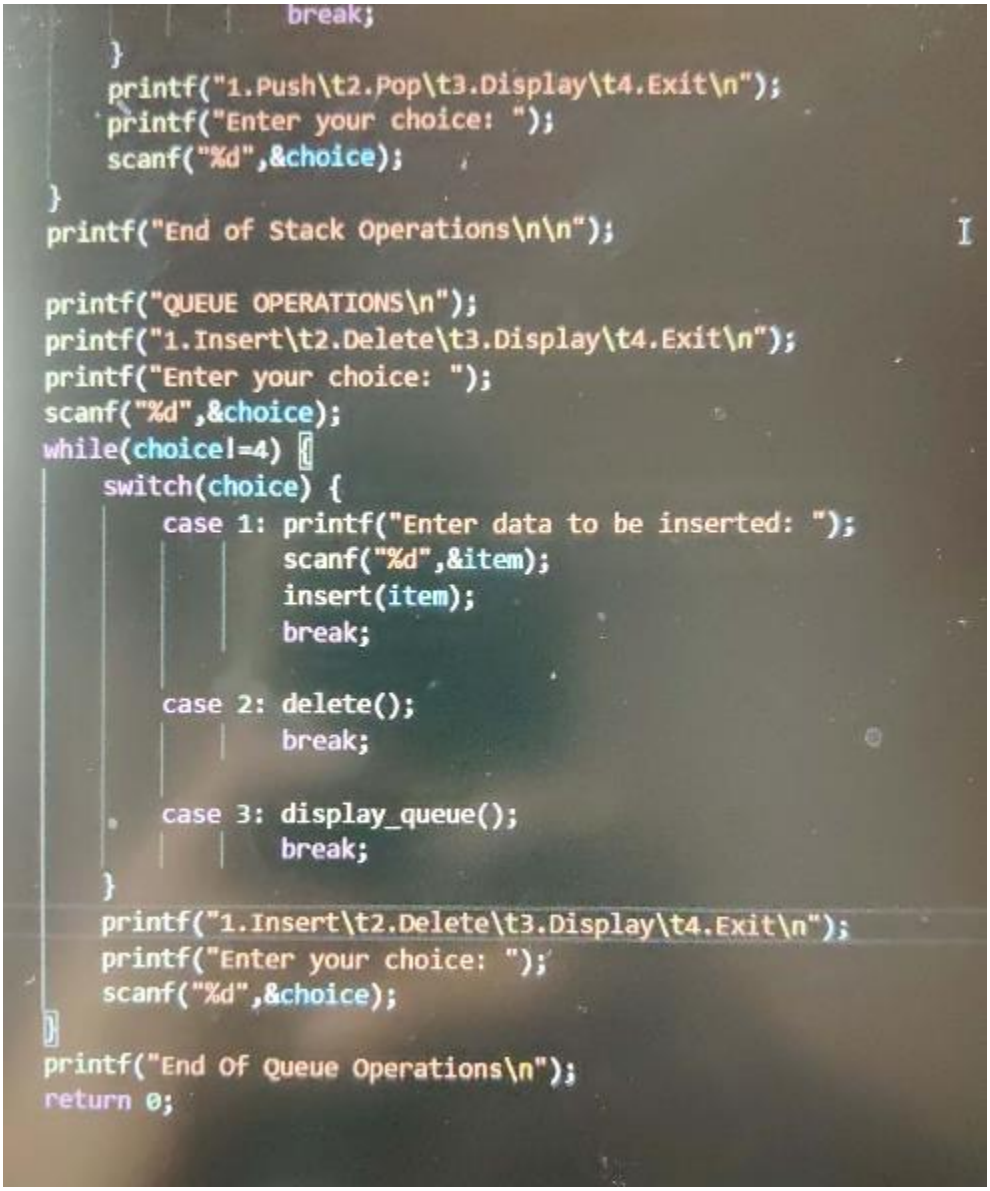
8.WAP to implement Stack & Queues using Linked Representation


Untitled-1.c
Untitled.c
C: > Users > dhruvam02 > OneDrive > Desktop > Untitled.c > main()
#include<stdio.h>
#include<stdlib.h>
struct node{
int data;
struct node*next;
};
struct node*front;
struct node*rear;
void push(struct node**top,int d) {
struct node*temp,n;
temp = (struct node*)malloc(sizeof(struct node));
if(temp == NULL) {
printf("Stack is full\n");
}
temp->data = d;
temp->next = *top;
*top = temp;
printf("%d is pushed\n",d);
}
void pop(struct node**top) {
struct node*temp;
if(*top==NULL) {
printf("Stack Underflow\n");
return;
}

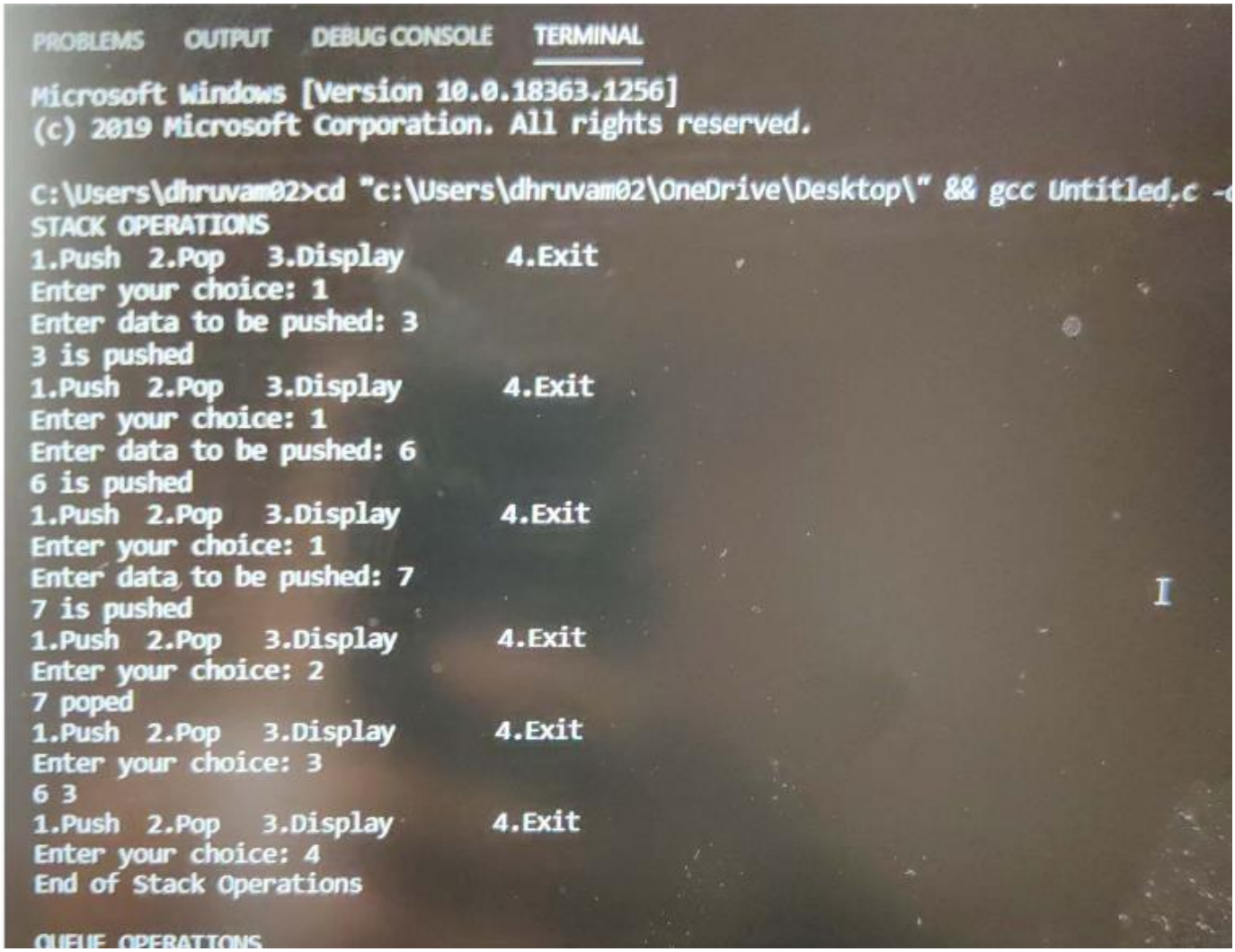

Untitled.c
dhruvam02 > OneDrive > Desktop > Untitled.c > main()
return;
}
temp = *top;
printf("%d poped\n",temp->data);
*top = (*top)->next;
free(temp);
void display(struct node* top) {
if(top == NULL){
printf("No Elements Present in Stack\n");
return;
}
while(top!=NULL) {
printf("%d ",top->data);
top = top->next;
}
printf("\n");
void insert(int d) {
struct node*n;
n = (struct node*)malloc(sizeof(struct node));
if(n == NULL){
printf("Queue Overflow\n");
return;
}
n->data = d;
if(front==NULL) {
front = n;

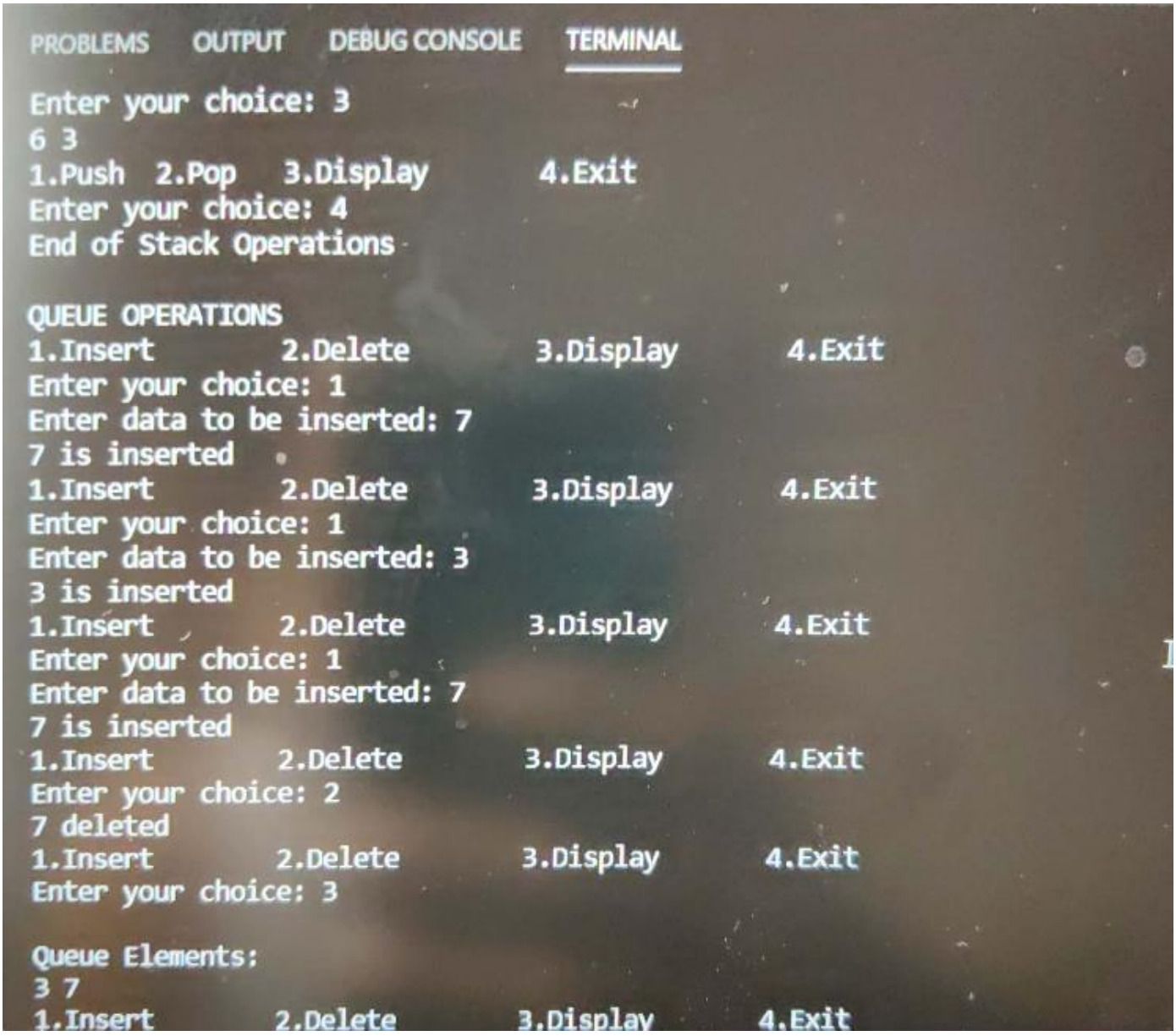

Untitled.c
dhruvam02 > OneDrive > Desktop > Untitled.c > main()
front = n;
rear = n;
front->next = NULL;
rear->next = NULL;
}
else {
rear->next = n;
rear = n;
rear->next = NULL;
}
printf("%d is inserted\n",d);
oid delete() {
struct node*temp;
if(front == NULL) {
printf("Queue Underflow\n");
return;
}
temp = front;
printf("%d deleted\n",temp->data);
front = front->next;
free(temp);
oid display_queue() {
struct node *temp;
temp = front;
if(front == NULL)
{
printf("\nEmpty queue\n");
}

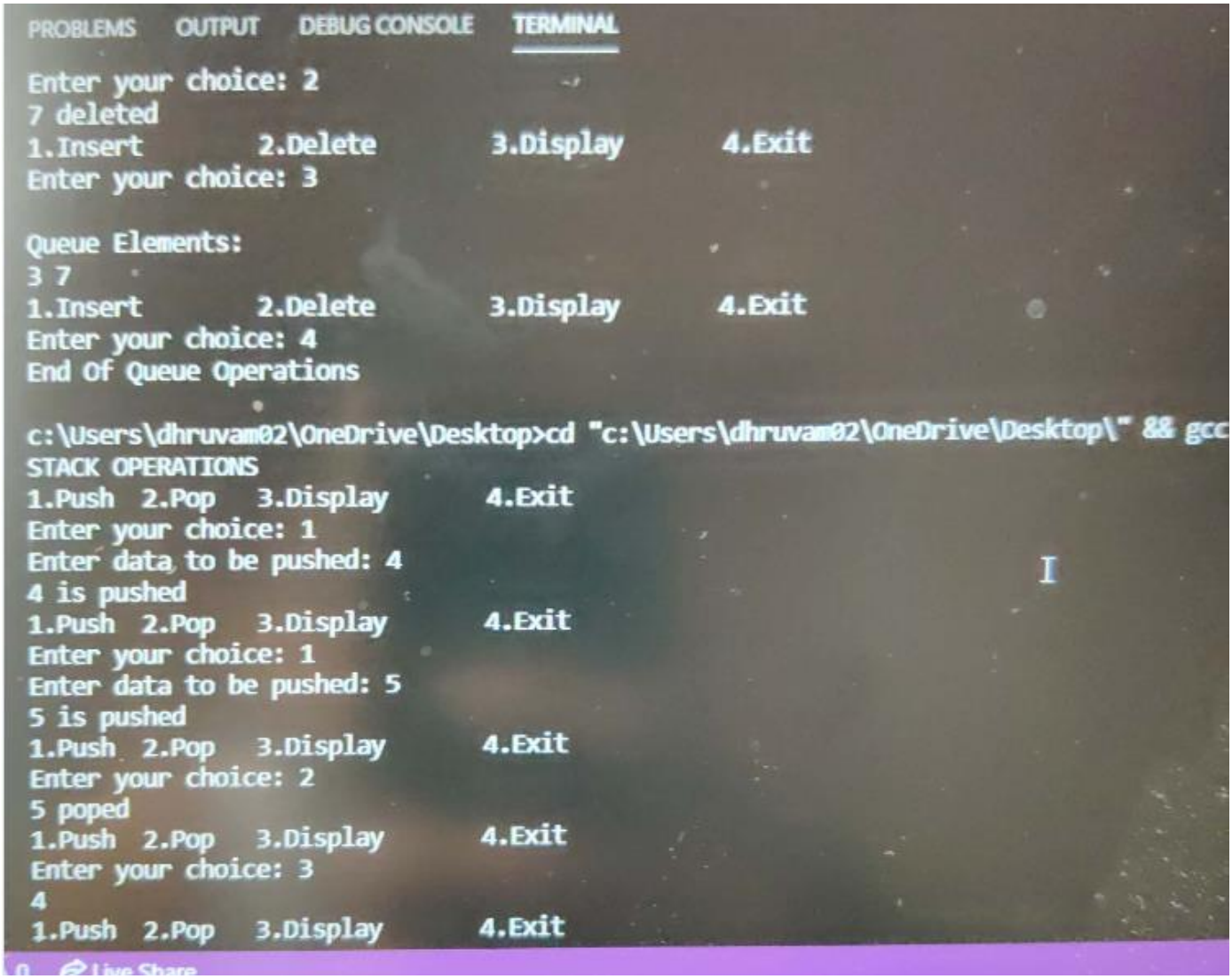

ers > dhruvam02 > OneDrive > Desktop > C Untitled.c > main()
else
{ printf("\nQueue Elements: \n");
while(temp != NULL)
{
printf("%d ",temp -> data);
temp = temp -> next;
}
printf("\n");
}
}
int main() {
struct node*stack = NULL;
printf("STACK OPERATIONS\n");
printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
int choice,item;
printf("Enter your choice: ");
scanf("%d",&choice);
while(choice!=4) {
switch(choice) {
case 1: printf("Enter data to be pushed: ");
scanf("%d",&item);
push(&stack,item);
break;
case 2: pop(&stack);
break;
case 3: display(stack);
break;
}
printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
printf("Enter your choice: ");

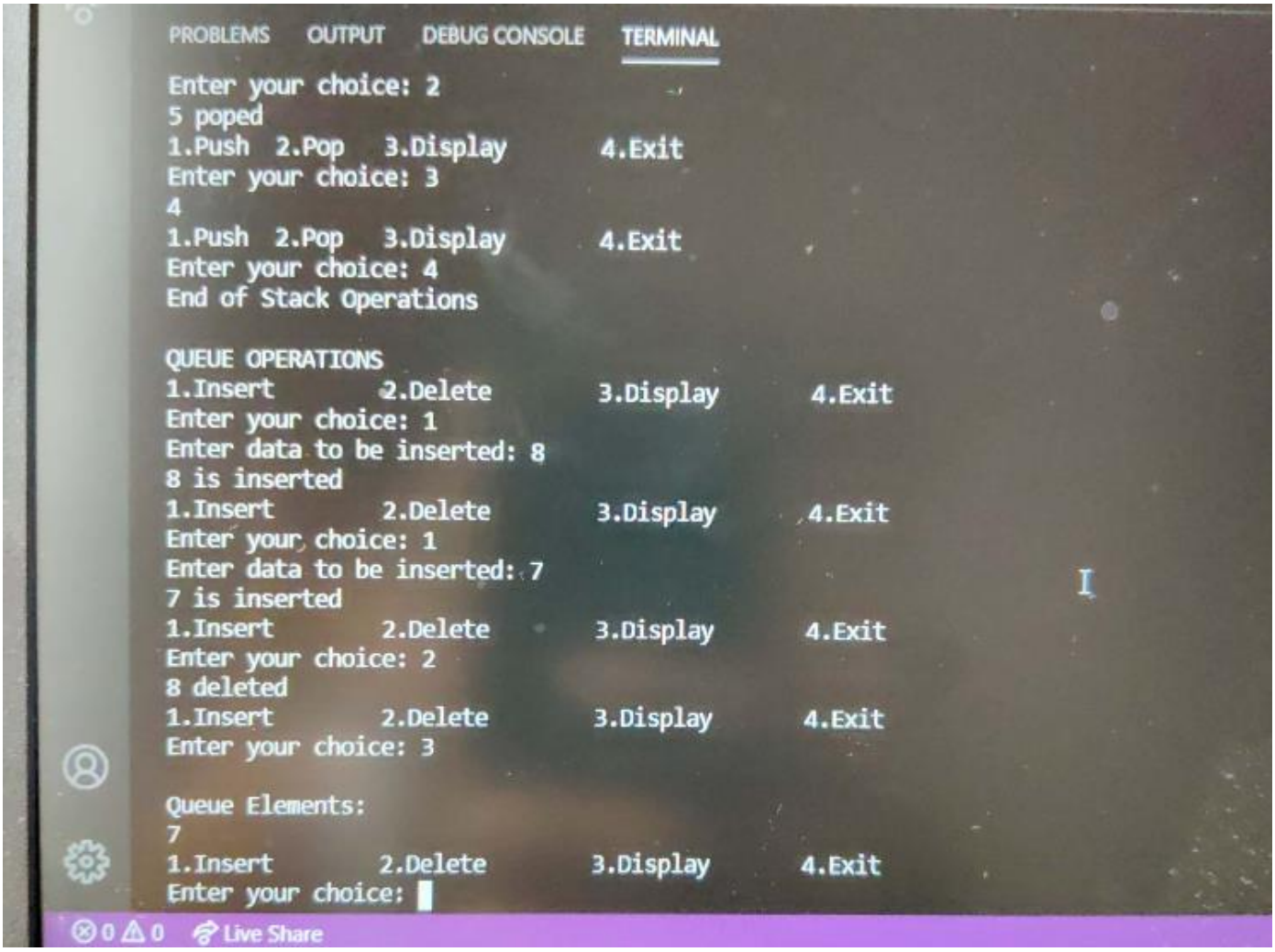


```
                break;
        }
        printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
    }
    printf("End of Stack Operations\n\n");

    printf("QUEUE OPERATIONS\n");
    printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
    printf("Enter your choice: ");
    scanf("%d",&choice);
    while(choice!=4) {
        switch(choice) {
            case 1: printf("Enter data to be inserted: ");
                    scanf("%d",&item);
                    insert(item);
                    break;

            case 2: delete();
                    break;

            case 3: display_queue();
                    break;
        }
        printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
    }
    printf("End Of Queue Operations\n");
    return 0;
```

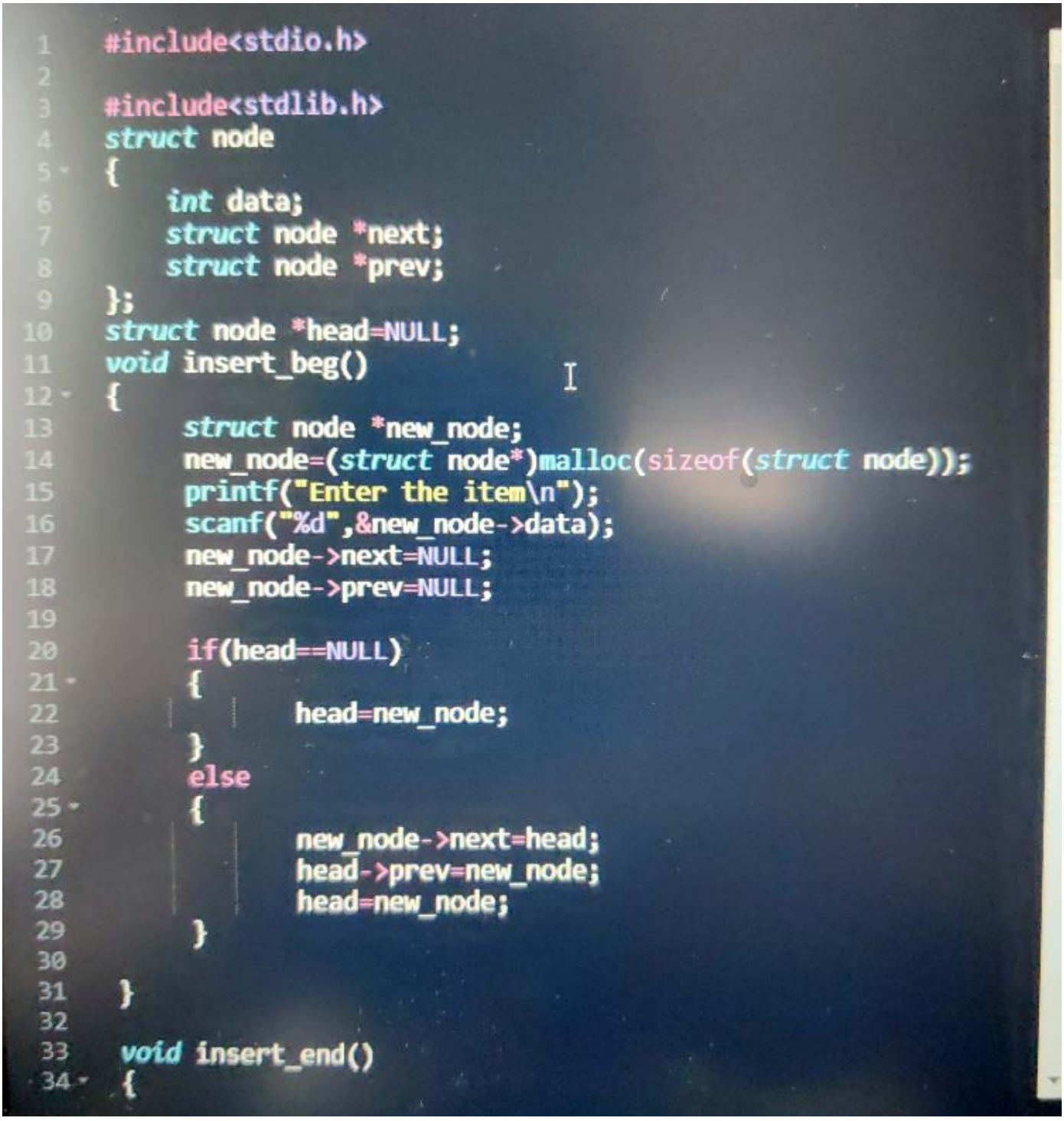

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Microsoft Windows [Version 10.0.18363.1256]
(c) 2019 Microsoft Corporation. All rights reserved.
C:\Users\dhruvam02>cd "c:\Users\dhruvam02\OneDrive\Desktop\" && gcc Untitled.c -
STACK OPERATIONS
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 3
3 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 6
6 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 7
7 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2
7 poped
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
6 3
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 4
End of Stack Operations
QUEUE OPERATIONS


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Enter your choice: 3
6 3
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 4
End of Stack Operations
QUEUE OPERATIONS
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter data to be inserted: 7
7 is inserted
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter data to be inserted: 3
3 is inserted
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 1
Enter data to be inserted: 7
7 is inserted
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 2
7 deleted
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 3
Queue Elements:
3 7
1.Insert 2.Delete 3.Display 4.Exit


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Enter your choice: 2
7 deleted
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 3
Queue Elements:
3 7
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 4
End Of Queue Operations
c:\Users\dhruvam02\OneDrive\Desktop>cd "c:\Users\dhruvam02\OneDrive\Desktop\" && gcc
STACK OPERATIONS
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 4
4 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 5
5 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2
5 poped
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
4
1.Push 2.Pop 3.Display 4.Exit
Live Share



9. WAP Implement doubly link list with primitive operations a) a) Create a doubly linked list. b) Insert a new node to the left of the node. b) c) Delete the node based on a specific value. c) Display the contents of the list

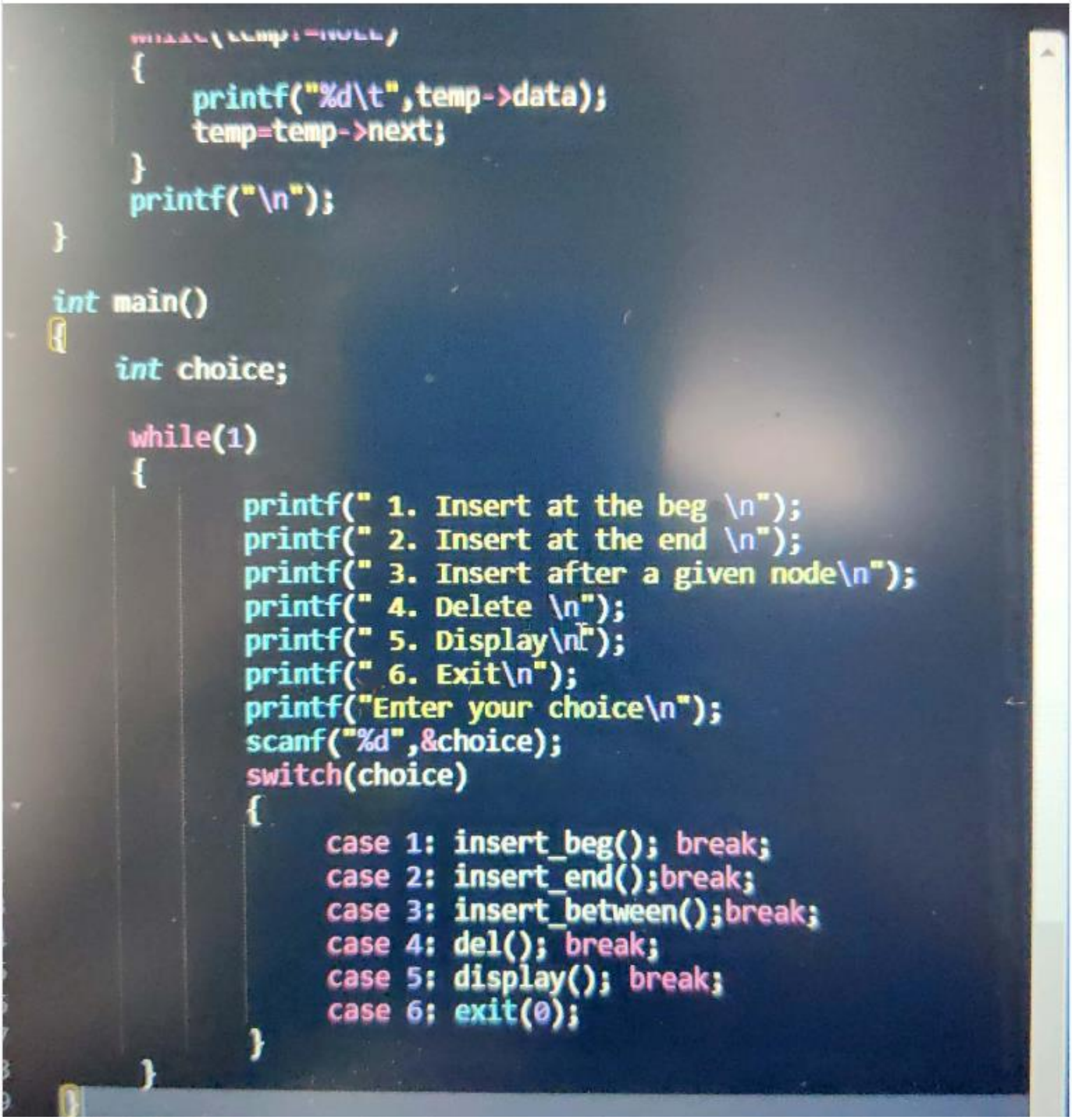


```c
#include<stdio.h>

#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
    struct node *prev;
};
struct node *head=NULL;
void insert_beg()
{
    struct node *new_node;
    new_node=(struct node*)malloc(sizeof(struct node));
    printf("Enter the item\n");
    scanf("%d",&new_node->data);
    new_node->next=NULL;
    new_node->prev=NULL;

    if(head==NULL)
    {
        head=new_node;
    }
    else
    {
        new_node->next=head;
        head->prev=new_node;
        head=new_node;
    }

}

void insert_end()
{
```

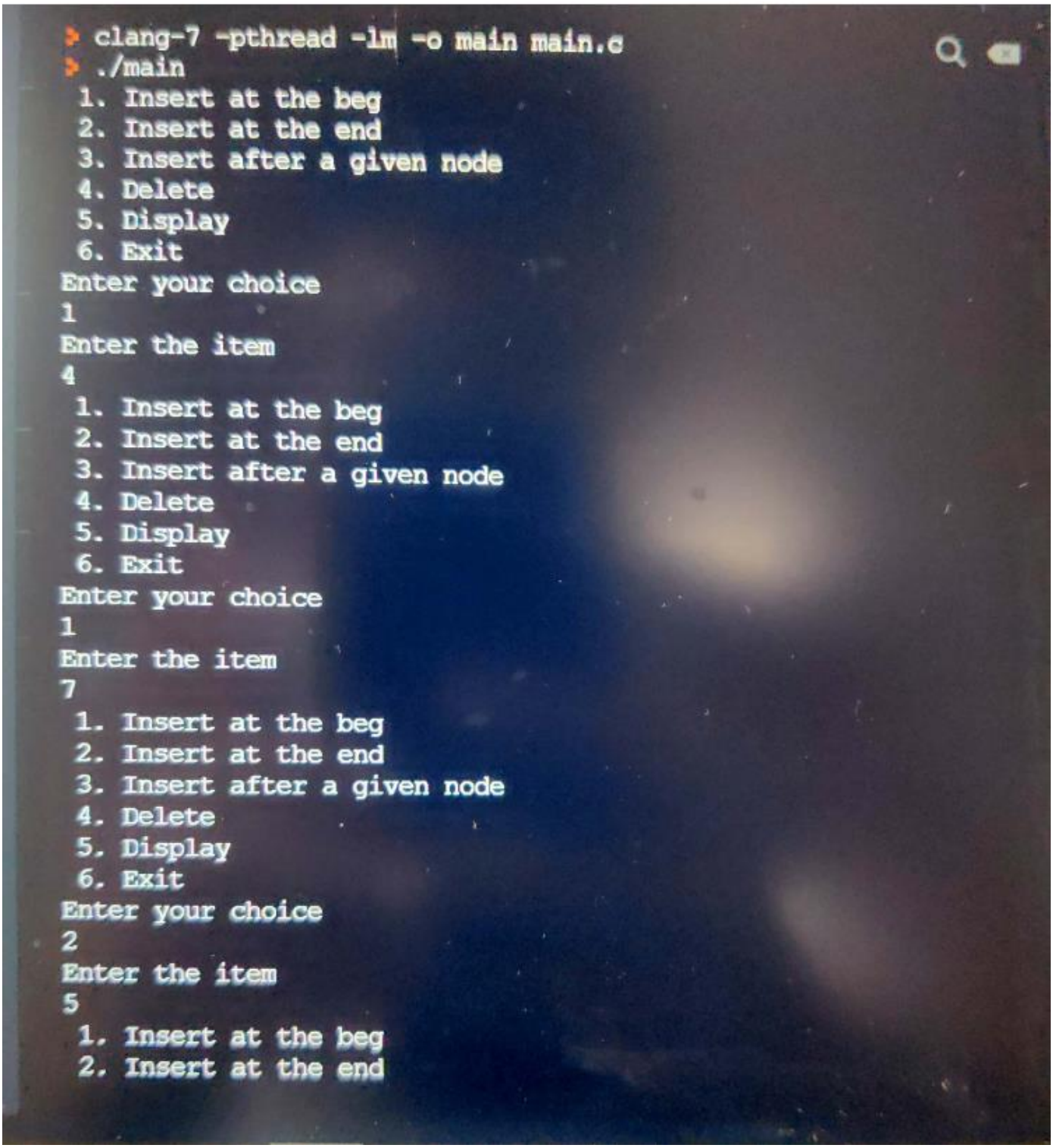
```
    struct node *new_node,*temp;
    new_node=(struct node*)malloc(sizeof(struct node));
    printf("Enter the item\n");
    scanf("%d",&new_node->data);
    new_node->next=NULL;
    new_node->prev=NULL;
    if(head==NULL)
    {
        head=new_node;
    }
    else
    {
        temp=head;
        while(temp->next!=NULL)
        temp=temp->next;
        temp->next=new_node;
        new_node->prev=temp;

    }

}
void insert_between()
{
    int listele;
    struct node *new_node,*temp;
    printf("Enter the element in the list\n");
    scanf("%d",&listele);
    new_node=(struct node*)malloc(sizeof(struct node));
    printf("Enter the new node data\n");
    scanf("%d",&new_node->data);
    new_node->next=NULL;
    new_node->prev=NULL;
    if(head==NULL)
```

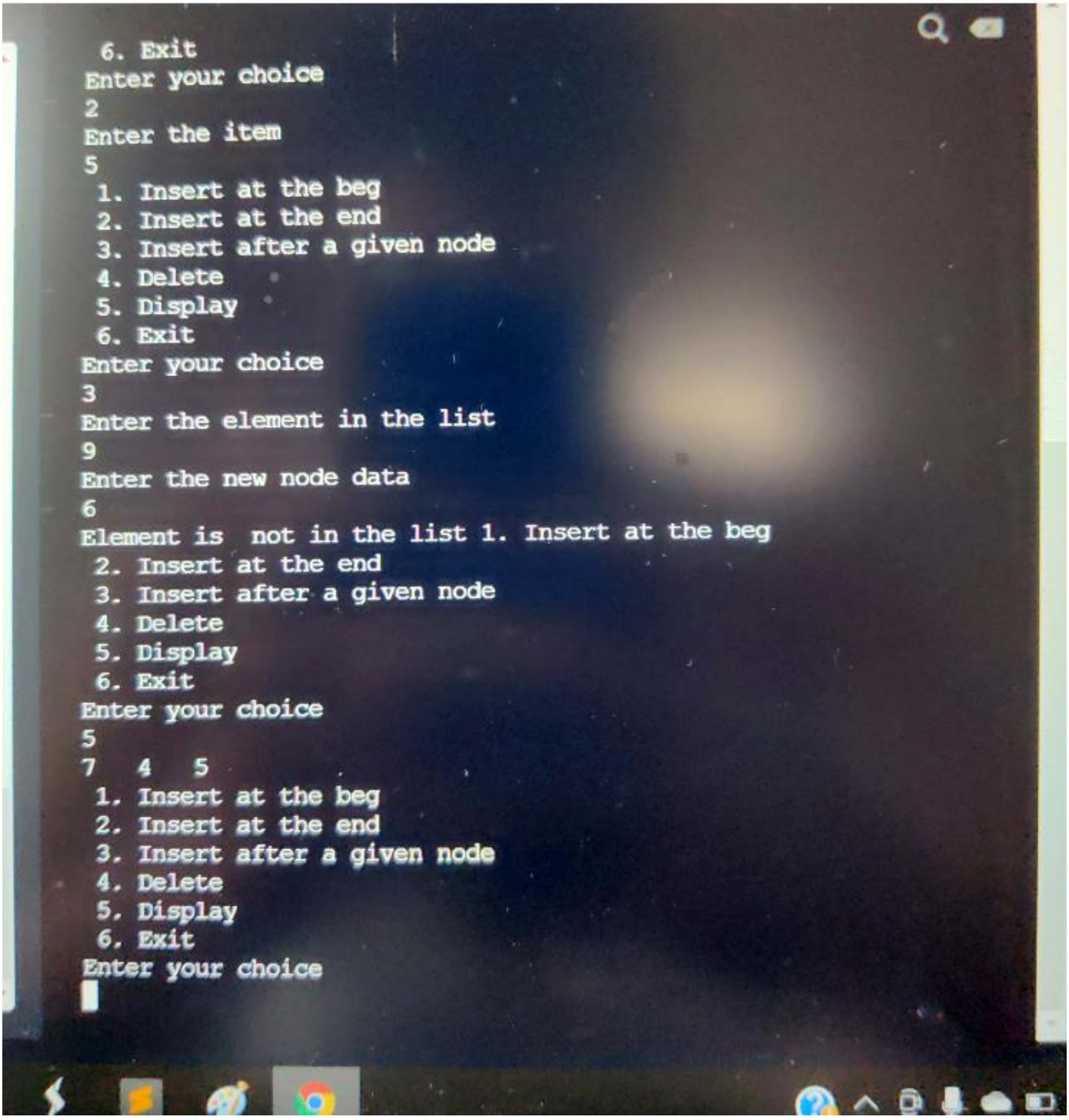
```
        if(head==NULL)
        {
            printf("Empty list\n"); return;
        }
        temp=head;
        while(temp->data!=listele)
        {
            temp=temp->next;
            if(temp==NULL)
            {
                printf("Element is  not in the list");
                return;
            }
        }
        new_node->next=temp->next;
        temp->next=new_node;
        new_node->prev=temp;
        new_node->next->prev=new_node;
    }
    void del()
    {
        struct node *temp;
        int ele;
        if(head==NULL)
        {
            printf("Empty List \n");
            return;
        }
        printf("Enter the element to be deleted\n");
        scanf("%d",&ele);
        temp=head;
        while(temp->data!=ele)
        {
            temp=temp->next;
```

```
    {
        temp=temp->next;
        if(temp==NULL)
        {
         printf("Element is not in the list\n");
         break;
        }
    }
    if(temp==head)
    {
        head=head->next;
    }
    else if(temp->next==NULL)
    {
            temp=temp->prev;
            temp->next=NULL;
    }

    else
    {
        temp->prev->next=temp->next;
        temp->next->prev=temp->prev;
    }
}
void display()
{
    struct node *temp;
    temp=head;
    while(temp!=NULL)
    {
        printf("%d\t",temp->data);
        temp=temp->next;
    }
    printf("\n");
```

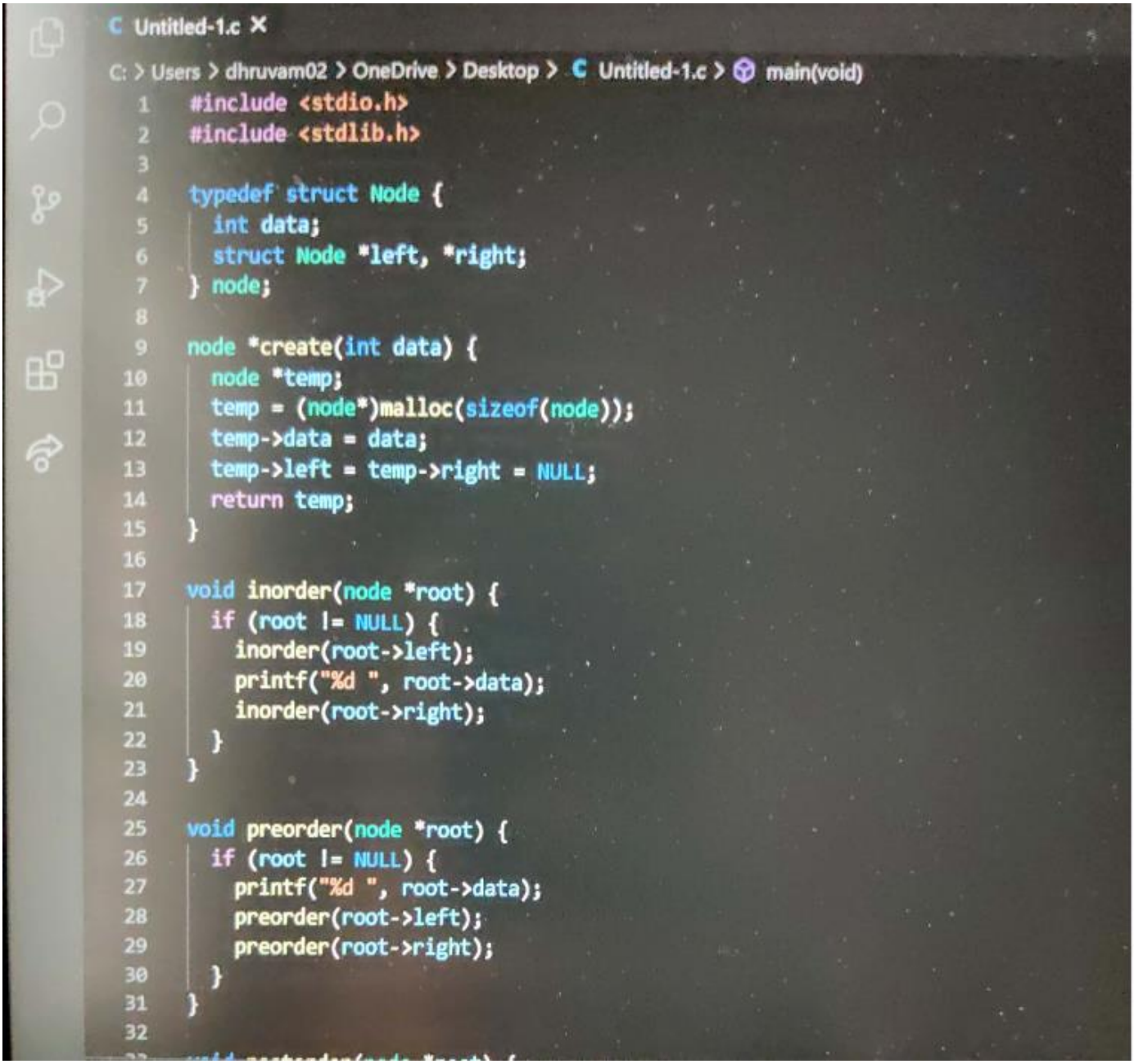


```
        {
            printf("%d\t",temp->data);
            temp=temp->next;
        }
        printf("\n");
}

int main()
{
    int choice;

    while(1)
    {
            printf(" 1. Insert at the beg \n");
            printf(" 2. Insert at the end \n");
            printf(" 3. Insert after a given node\n");
            printf(" 4. Delete \n");
            printf(" 5. Display\n");
            printf(" 6. Exit\n");
            printf("Enter your choice\n");
            scanf("%d",&choice);
            switch(choice)
            {
                case 1: insert_beg(); break;
                case 2: insert_end();break;
                case 3: insert_between();break;
                case 4: del(); break;
                case 5: display(); break;
                case 6: exit(0);
            }
    }
```

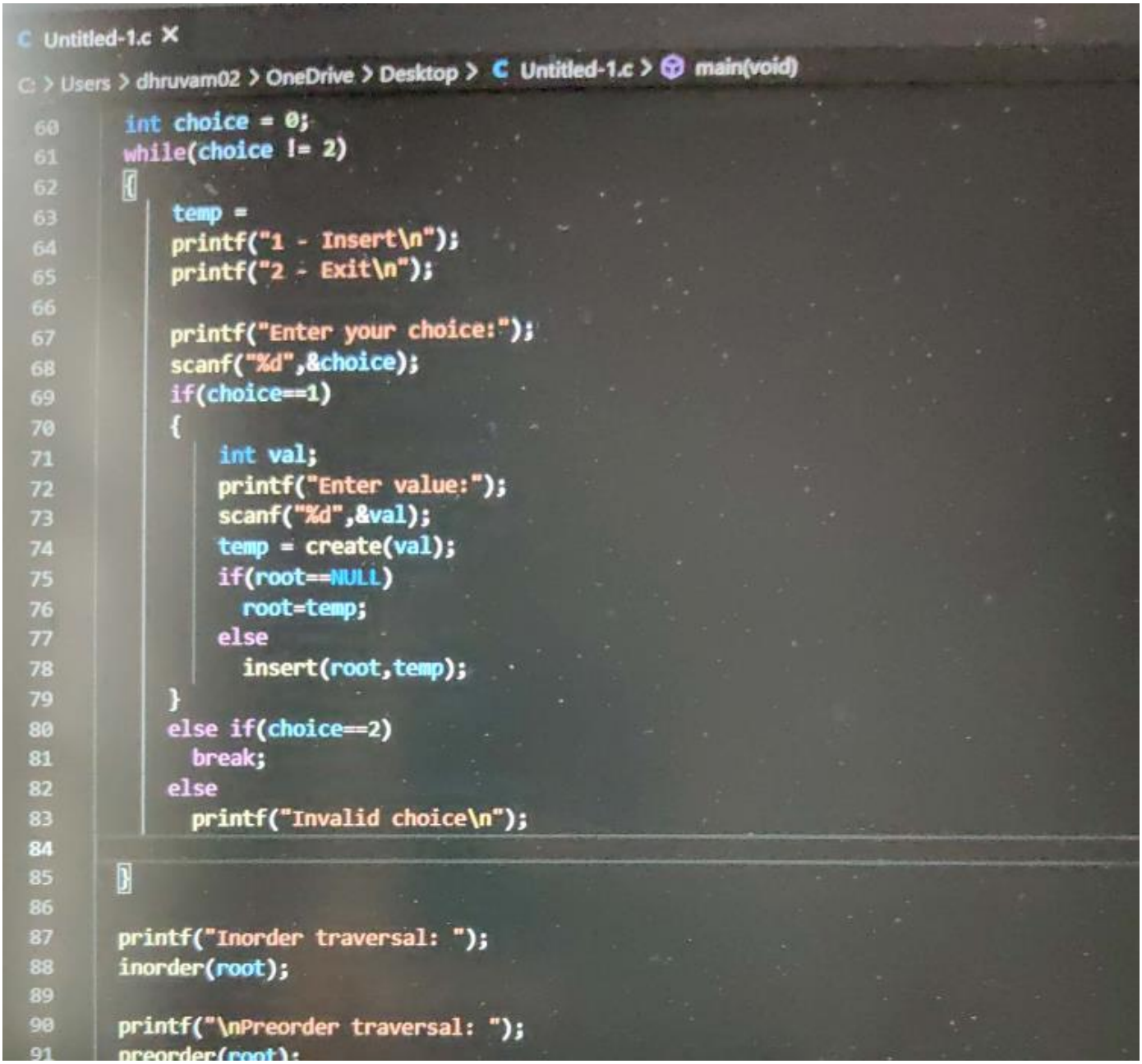

clang-7 -pthread -lm -o main main.c
./main
1. Insert at the beg
2. Insert at the end
3. Insert after a given node
4. Delete
5. Display
6. Exit
Enter your choice
1
Enter the item
4
1. Insert at the beg
2. Insert at the end
3. Insert after a given node
4. Delete
5. Display
6. Exit
Enter your choice
1
Enter the item
7
1. Insert at the beg
2. Insert at the end
3. Insert after a given node
4. Delete
5. Display
6. Exit
Enter your choice
2
Enter the item
5
1. Insert at the beg
2. Insert at the end

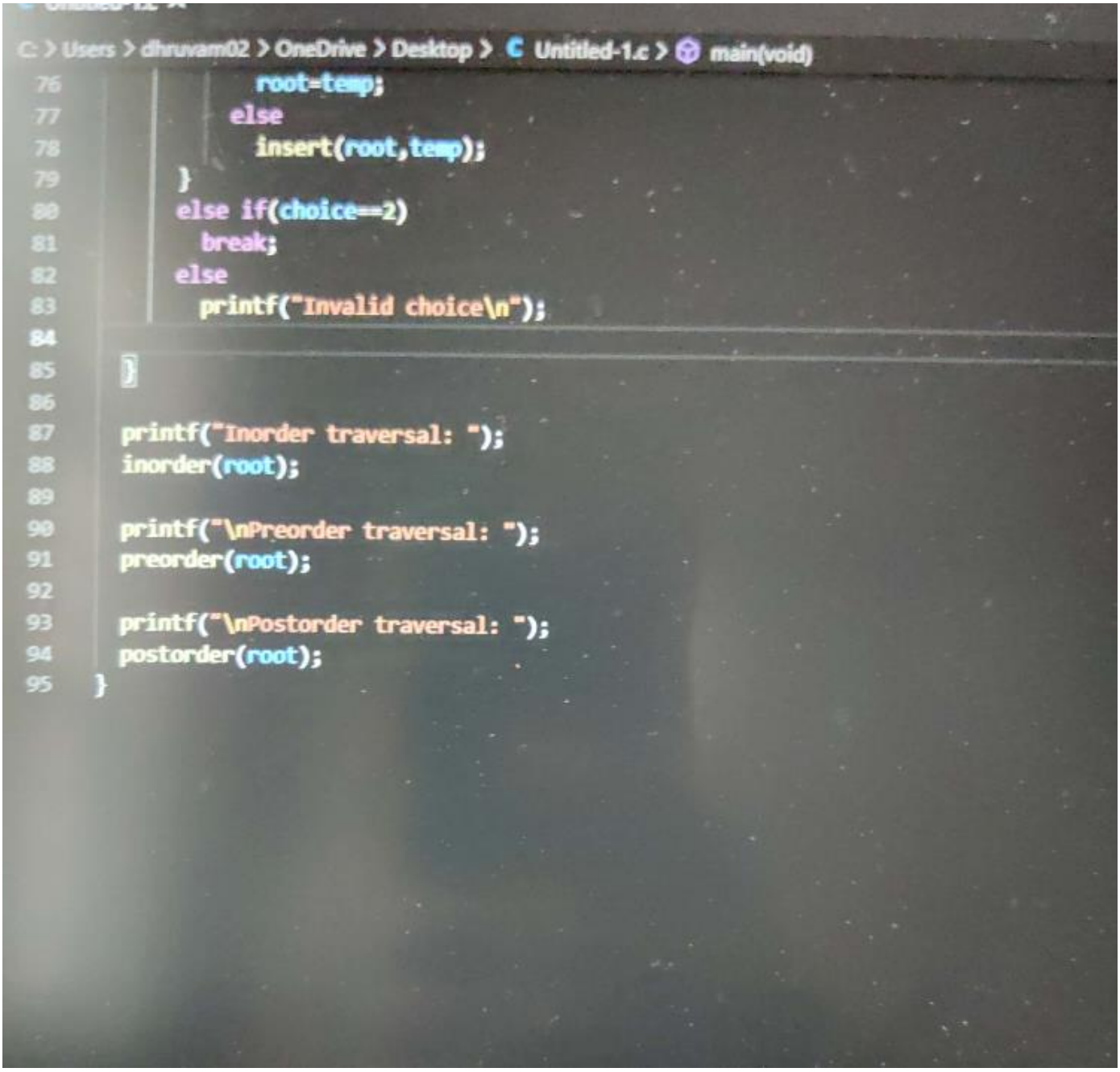


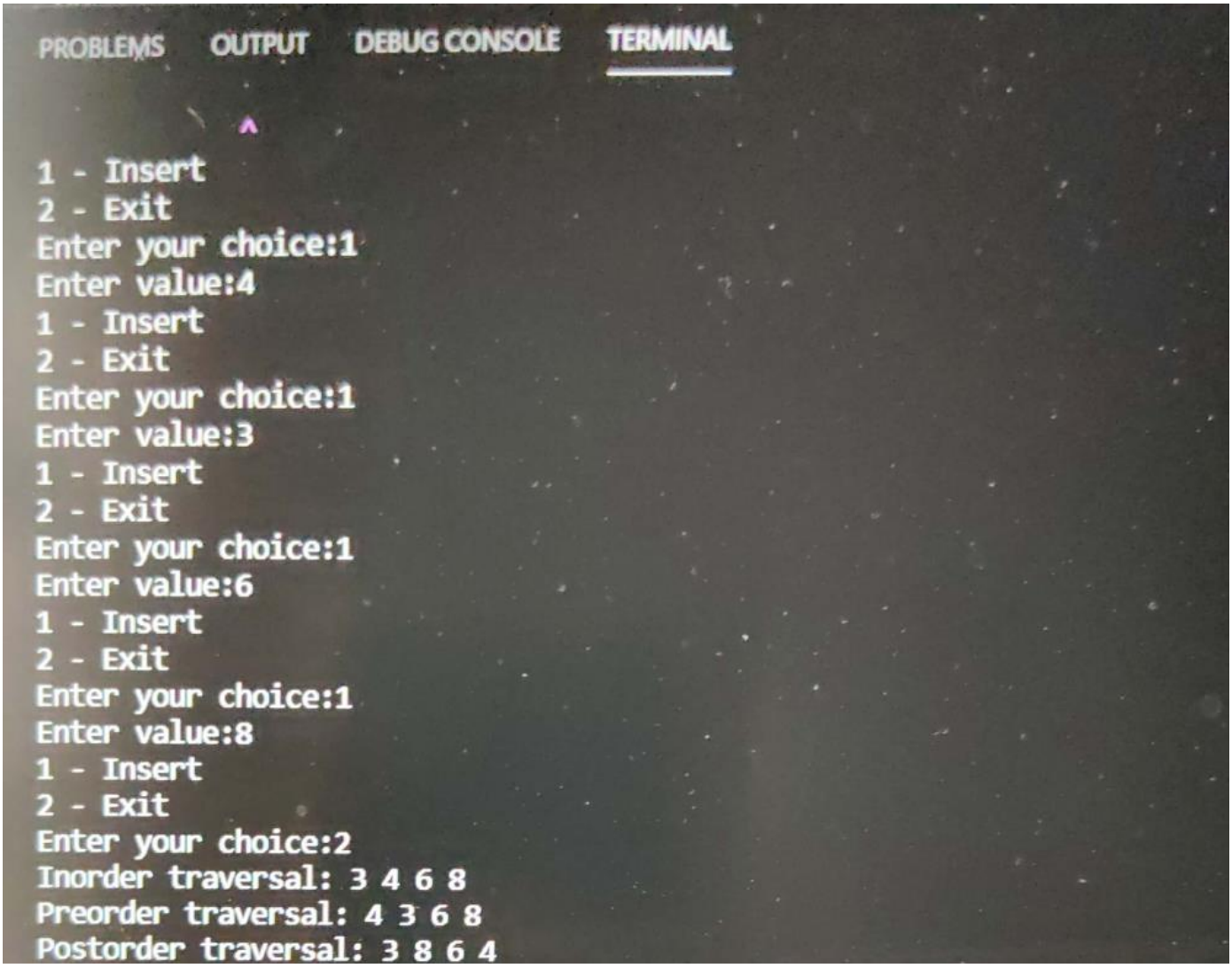
10.Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, preorder and post order c) To display the elements in the tree.

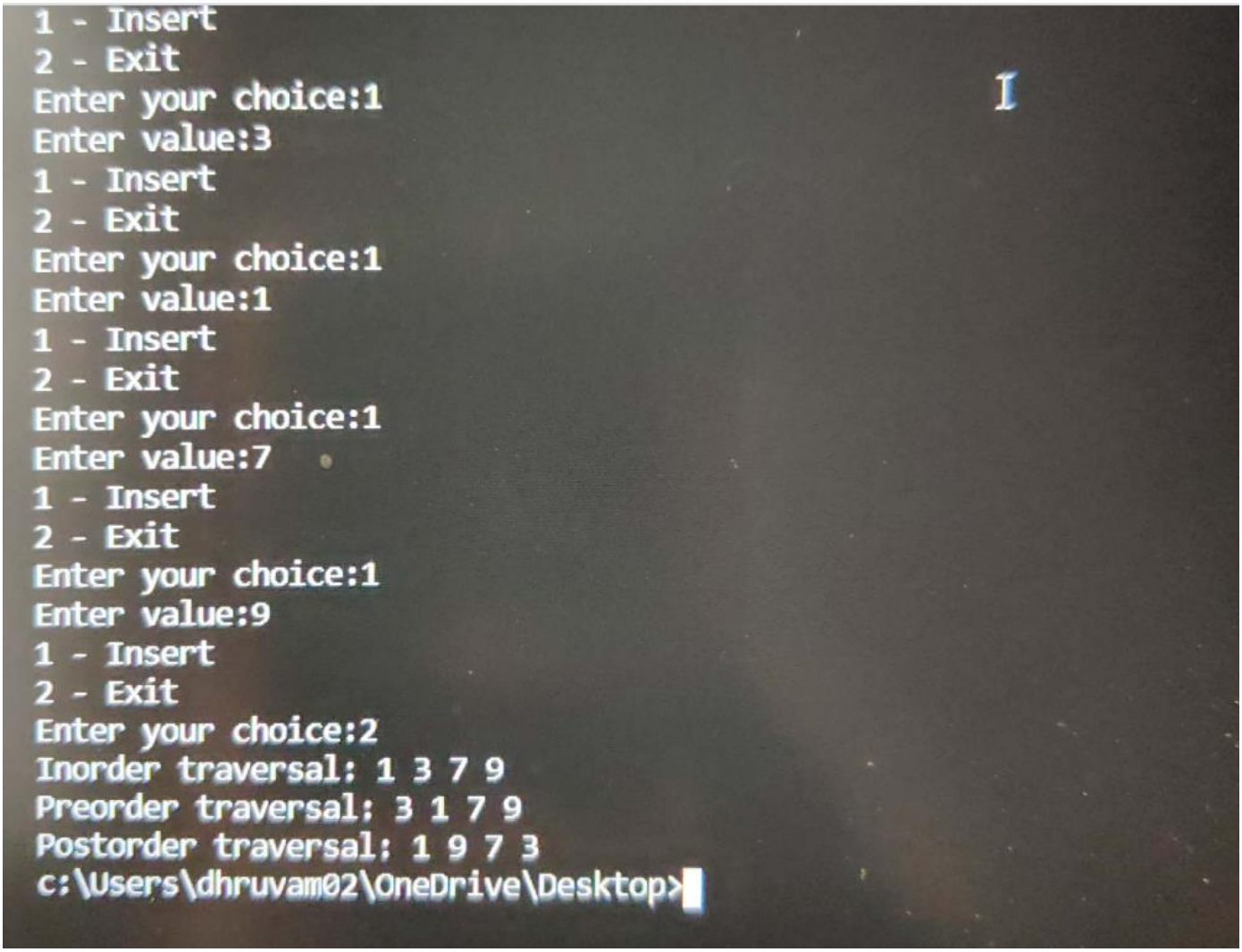

Untitled-1.c
C: > Users > dhruvam02 > OneDrive > Desktop > Untitled-1.c > main(void)
#include <stdio.h>
#include <stdlib.h>
typedef struct Node {
int data;
struct Node *left, *right;
} node;
node *create(int data) {
node *temp;
temp = (node*)malloc(sizeof(node));
temp->data = data;
temp->left = temp->right = NULL;
return temp;
}
void inorder(node *root) {
if (root != NULL) {
inorder(root->left);
printf("%d ", root->data);
inorder(root->right);
}
}
void preorder(node *root) {
if (root != NULL) {
printf("%d ", root->data);
preorder(root->left);
preorder(root->right);
}
}

```
31  }
32
33  void postorder(node *root) {
34    if (root != NULL) {
35      postorder(root->left);
36      postorder(root->right);
37      printf("%d ", root->data);
38    }
39  }
40
41  void insert(node *root, node *temp) {
42    if(temp->data<root->data){
43      if(root->left!=NULL)
44          insert(root->left,temp);
45      else
46          root->left = temp;
47    }
48    if(temp->data>root->data)
49    {
50        if(root->right!=NULL)
51          insert(root->right,temp);
52        else
53          root->right=temp;
54    }
55  }
56
57  int main(void) {
58    node *root = NULL,*temp;
59
60    int choice = 0;
61    while(choice != 2)
```


Untitled-1.c
C: > Users > dhruvam02 > OneDrive > Desktop > Untitled-1.c > main(void)
int choice = 0;
while(choice != 2)
{
temp =
printf("1 - Insert\n");
printf("2 - Exit\n");
printf("Enter your choice:");
scanf("%d",&choice);
if(choice==1)
{
int val;
printf("Enter value:");
scanf("%d",&val);
temp = create(val);
if(root==NULL)
root=temp;
else
insert(root,temp);
}
else if(choice==2)
break;
else
printf("Invalid choice\n");
}
printf("Inorder traversal: ");
inorder(root);
printf("\nPreorder traversal: ");


C: > Users > dhruvam02 > OneDrive > Desktop > Untitled-1.c > main(void)
76 root=temp;
77 else
78 insert(root,temp);
79 }
80 else if(choice==2)
81 break;
82 else
83 printf("Invalid choice\n");
84
85 }
86
87 printf("Inorder traversal: ");
88 inorder(root);
89
90 printf("\nPreorder traversal: ");
91 preorder(root);
92
93 printf("\nPostorder traversal: ");
94 postorder(root);
95 }


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
1 - Insert
2 - Exit
Enter your choice:1
Enter value:4
1 - Insert
2 - Exit
Enter your choice:1
Enter value:3
1 - Insert
2 - Exit
Enter your choice:1
Enter value:6
1 - Insert
2 - Exit
Enter your choice:1
Enter value:8
1 - Insert
2 - Exit
Enter your choice:2
Inorder traversal: 3 4 6 8
Preorder traversal: 4 3 6 8
Postorder traversal: 3 8 6 4


1 - Insert
2 - Exit
Enter your choice:1
Enter value:3
1 - Insert
2 - Exit
Enter your choice:1
Enter value:1
1 - Insert
2 - Exit
Enter your choice:1
Enter value:7
1 - Insert
2 - Exit
Enter your choice:1
Enter value:9
1 - Insert
2 - Exit
Enter your choice:2
Inorder traversal: 1 3 7 9
Preorder traversal: 3 1 7 9
Postorder traversal: 1 9 7 3
c:\Users\dhruvam02\OneDrive\Desktop>