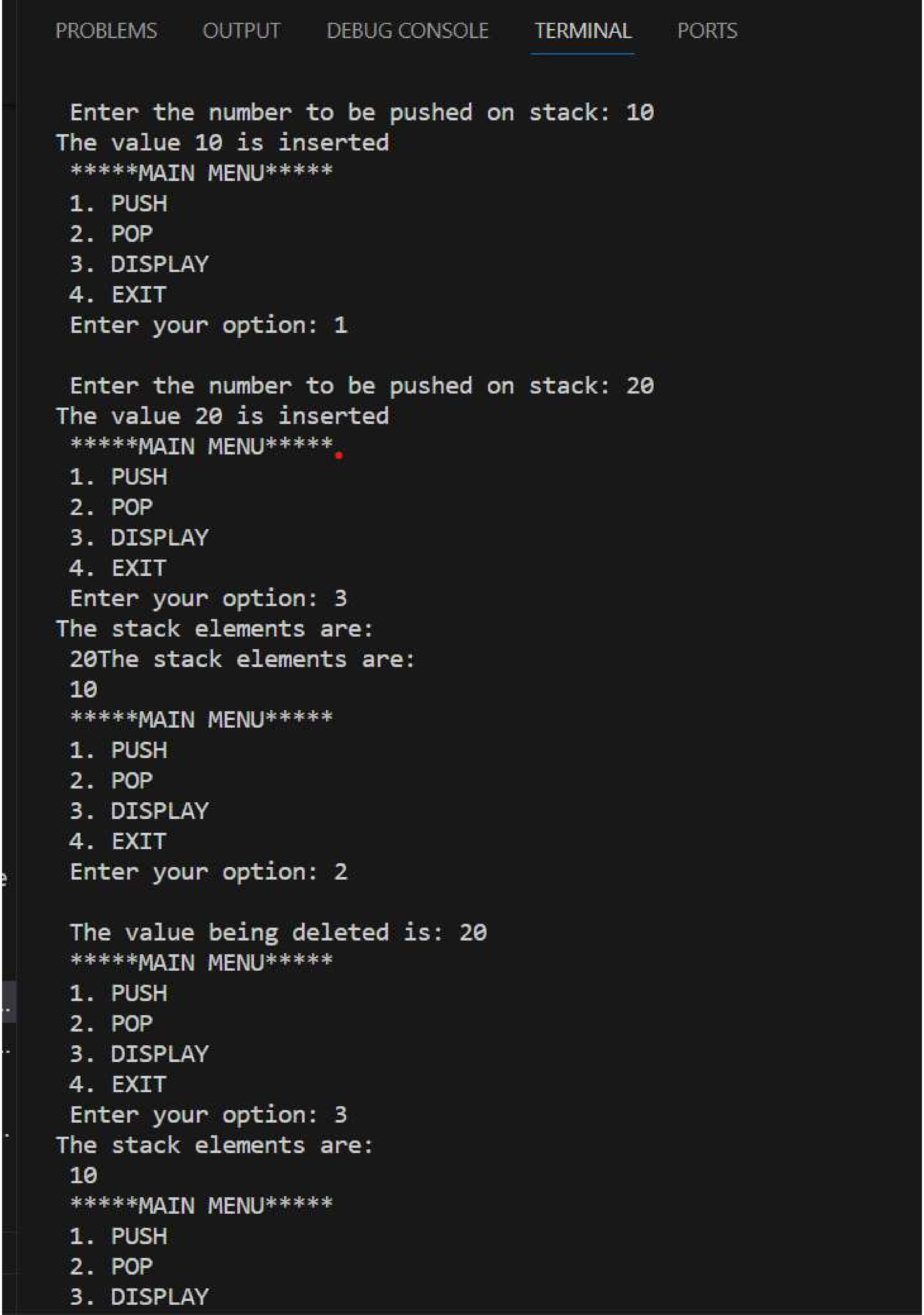


```
 Enter the number to be pushed on stack: 10
The value 10 is inserted
 *****MAIN MENU*****
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option: 1

 Enter the number to be pushed on stack: 20
The value 20 is inserted
 *****MAIN MENU*****
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option: 3
The stack elements are:
 20The stack elements are:
 10
 *****MAIN MENU*****
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option: 2

 The value being deleted is: 20
 *****MAIN MENU*****
 1. PUSH
 2. POP
 3. DISPLAY
 4. EXIT
 Enter your option: 3
The stack elements are:
 10
 *****MAIN MENU*****
 1. PUSH
 2. POP
 3. DISPLAY
```

```
struct stack *push (struct stack *top, int val) {
struct stack *ptr;
ptr= (struct stack*) malloc (sizeof(struct stack));
ptr -> data = val ;
 if (top== NULL ){
    ptr-> next = NULL;
    top= ptr;
    printf ("The value %d inserted is", val ); }
 else {
   ptr->next = top;
   top= ptr;
   printf ("The value %d is inserted", val );
 }
 return top;
}

struct stack * display (struct stack * top) {
  struct stack * ptr;
  ptr = top;
   if (top== NULL)
    printf ("\n Stack is empty");
  else {
   while (ptr != NULL) {
     printf ("the stack elements are: ");
     printf ("\n %d", ptr -> data );
     ptr = ptr -> next;
  }
 }
return top;
 }
```

Excd
[illegible]
29/11/24

LAB-5

29-1-24

2(a) Stack implementation using single linked list

```
# include <stdio.h>
# include <stdlib.h>
struct stack {
      int data
   struct stack* next;
 };
struct stack * top= NULL;
struct stack* push (struct stack*, int);
struct stack* display (struct stack *);
struct stack *pop (struct stack *);
void main () {
      int val, option;
      while (1) {
      printf(" 1) Push \n 2) POP \n 3) DISPLAY \n 4) EXIT");
      printf ("\n Enter your option: ");
      Scanf ("%d", & option);
      switch (option) {
        case 1:
        printf ("\n Enter the no. to be pushed on stack");
        Scanf ("%d", & val);
        top= push (top, val);
        break;
       case 2:
       top= pop (top) ;
       break;
       case 3:
       top= display (top);
       break;
      case 4: exit (0);
     default: printf ("Invalid input");
  }
 }
}
```

```
struct stack* pop (struct stack* top)
{
        struct stack * ptr;
        ptr= top;
        if (top== NULL);
        printf ("\n Stack ~~overflow~~ underflow ");
    else {
        top= top->next;
        printf ("\n The value being deleted is : %d", ptr->data);
        free(ptr) ;
    }
    return top;
}
```

2 (b) Queue implementation using single linked list.

```
# include <stdio.h>
# include <stdlib.h>
struct node {
    int data ;
    struct node * next;
};
  struct queue {
  struct node* front;
  struct node* rear; };

struct queue * createQueue () {
    struct queue* q=(struct queue*) malloc
                    (sizeof (struct queue));
    q-> front = q-> rear = NULL;
    return q;
}
```

```
    q-> front = ptr;
    q-> rear = ptr;
    q-> front -> next = q-> rear -> next = NULL;
  }
  else {
    q-> rear -> next = ptr;
    q -> rear = ptr;
    q -> rear -> next = NULL; }
  return q; }

struct queue * display (struct queue *q)
{   struct node *ptr;
    ptr = q-> front;
    if (ptr= =NULL)
      printf ("\n Queue is empty \n");
   else {
     printf ("\n")
   while (ptr != q -> rear)
    {
     printf ("%d\t", ptr ->data);
     ptr =ptr->next;
     }
    printf ("%d \t", ptr->data); }
         return q;
  }
struct queue *delete_element (struct queue *q)
{
  struct node * ptr;
 ptr= q-> front;
 if (q-> front == NULL)
  printf ("\n Underflow \n");
```

```
struct queue *q;
struct queue* insert (struct queue*, int );
struct queue* delete_element (struct queue *);
struct queue* display (struct queue *);
void main () {
    int val, option;
    q= create_queue(q);
    while (1) {
    printf ("1) insert\n2) Delete \n 3) Display \n 4) Exit");
    printf ("Enter your option : ");
    scanf ("%d", & option;
   switch (option) {
    case 1:
    printf ("\n Enter the number to insert in the
                  queue : ");
    scanf ("%d", & val);
     q= insert (q, val );
     printf ("\n The value %d is inserted in queue", val
     break;
    case 2:    q= delete_element (q) ; break;
    case 3:   q= display (q); break;
    case 4:   exit (0);
   default : printf (" Invalid input" );
  }
 }
}
struct queue* insert (struct queue *q, int val ){
    struct node* ptr;
  ptr= (struct node*)malloc (sizeof (struct n
  ptr -> data = val;
 if (q-> front == NULL) {
```

Date
Page

```
else {
q->front = q->front -> next;
printf ("\n The value being deleted is : %d\n",
        ptr-> data);
  free (ptr); }
return q;
}
```

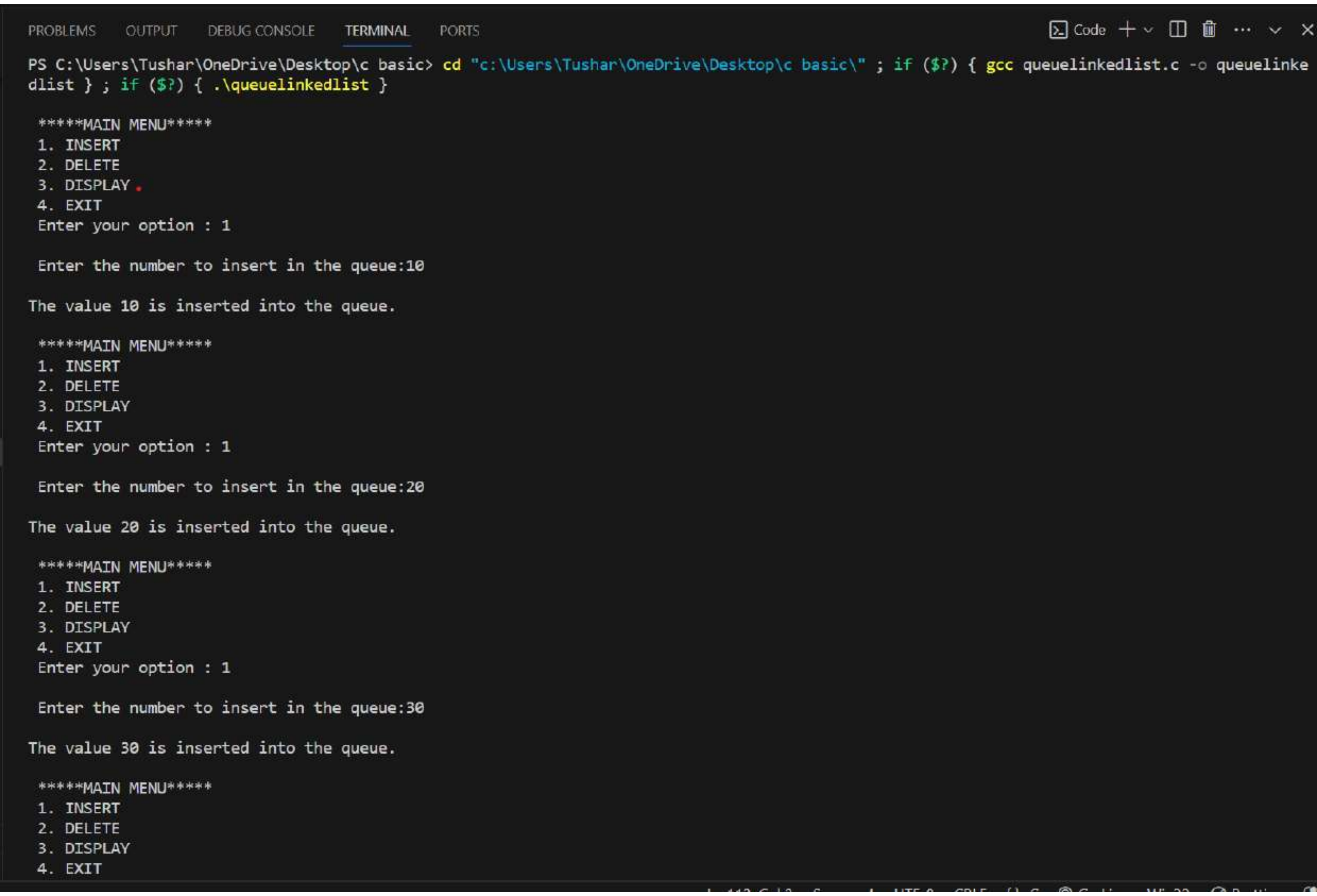
(1) Perform sort, reverse, concatenation on single linked list:

```
-> #include <stdio.h>
#include <stdlib.h>
struct Node {
     int data;
     struct Node* next;
};
void insertAtBeginning( struct Node **head, int data ){
     struct Node* newNode= (struct Node* )malloc (sizeof
                              (struct Node));
     newnode -> data = data;
     newnode -> nex = *head;
     *head = newNode.
}
void printList (struct Node* head) {
          while (head! =NULL) {
               printf ("%d", head-> data);
               head = head -> next ; }
          printf ("\n");
}
```

Emitted
NA
29/1/24

```
void sortList (struct Node** head) {
    struct Node* current, *nextNode;
    int temp;
    current = *head;
    while ( current != NULL) {
        nextNode = current ->next;
        while (nextNode != NULL) {
            if (current ->data > nextNode ->data ){
                temp= current ->data;
                current ->data= nextNode ->data;
                nextNode-> data = temp;
            }
            nextNode = nextNode ->next;
        }
    }
}
void reverseList (struct Node ** head) {
    struct Node * prev, * current, *nextNode;
    prev = NULL;
    current= *head;
    while (current != NULL) {
        nextNode = current ->next;
        current -> next = prev;
        prev = current ;
        current = nextNode;
    }
    *head = prev;
}
void concatenateLists (struct Node ** list1, struct Node*
                list2) {
    if (* list 1 = = NULL ) {
        * list1 = list 2;
        return;
    }
```

```
struct Node* temp = *list 1 ;
 while (temp->next ! = NULL) {
         temp= temp->next; }
      temp->next = list 2;
}
void  main() {
      struct Node* list 1 = NULL;
      struct Node* list 2 = NULL;
      int choice;
      int data;
while(1) {
   printf ("(1)Insert into list 1 \n");
   printf ("(2)Insert into list 2 \n");
   printf ("(3)Sort list \n");
    printf ("(4) Reverse list 2\n");
  printf (" 5) concatenate lists \n");
  printf (" (6) Print lists \n");
  printf (7. Exit \n");
printf ("Enter your choice : ");
 Scanf ("%d", &choice);
 switch (choice ){
      case 1:
       printf ("Enter data to insert into list 1: ");
       Scanf ("%d", & data);
       insertAtBeginning (& list1, data);
         break;
    case 2:
       printf ("Enter data to insert into list 2: ");
       Scanf ("%d", & data);
       insertAtBeginning (& list 2, data);
       break;
case 3:
  sortList(&list1);
  printf("List 1 is sorted.\n");
    break;
case 4:
    reverseList(&list1);
    printf("List 1 is reversed.\n");
     break;
 case 5:
     concatenateLists(&list1, list2);
     printf("Lists concatenated.\n");
      break;
case 6:
     printf("List 1: ");
     printList(list1);
     printf("List 2: ");
     printList(list2);
     break;
case 7:
    exit(0);
    break;
  default:
       printf("Invalid choice \n");
 }
}
}
```



```
PS C:\Users\Tushar\OneDrive\Desktop\c basic> cd "c:\Users\Tushar\OneDrive\Desktop\c basic\" ; if ($?) { gcc queuelinkedlist.c -o queuelinke
dlist } ; if ($?) { .\queuelinkedlist }

 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. DISPLAY
 4. EXIT
 Enter your option : 1

 Enter the number to insert in the queue:10

The value 10 is inserted into the queue.

 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. DISPLAY
 4. EXIT
 Enter your option : 1

 Enter the number to insert in the queue:20

The value 20 is inserted into the queue.

 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. DISPLAY
 4. EXIT
 Enter your option : 1

 Enter the number to insert in the queue:30

The value 30 is inserted into the queue.

 *****MAIN MENU*****
 1. INSERT
 2. DELETE
 3. DISPLAY
 4. EXIT
```