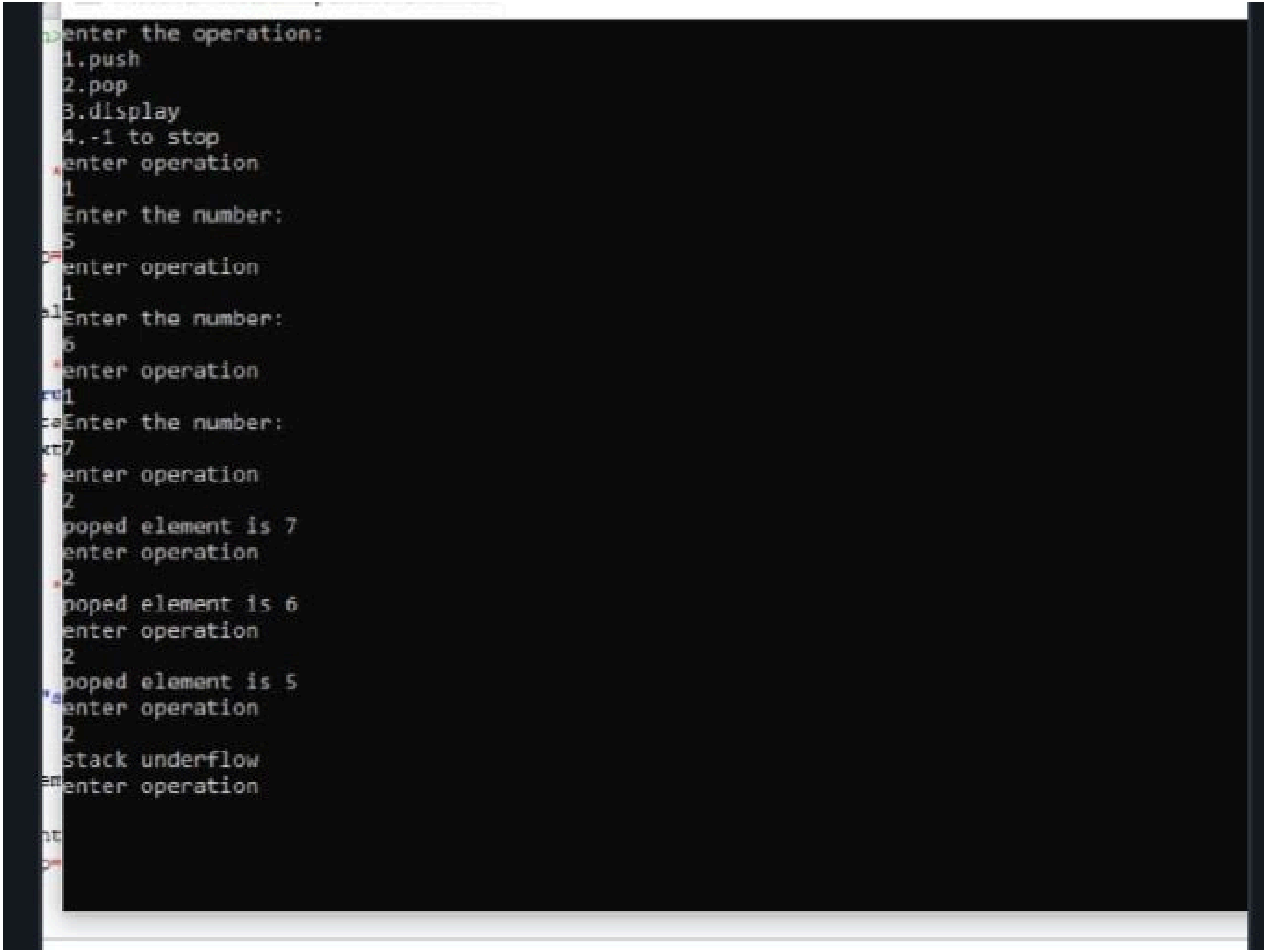
Stack Implementation;

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

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

struct Stack {
        struct Node* top;
};

int is_empty(struct Stack* stack){
      return stack -> top = = NULL;
}

Void push(struct Stack* stack, int data) {
  struct Node* new_node = (struct Node*) malloc
                    (size of(struct Node));
   if (! new_node) {
        pf ("Memory allocation error.\n");
        return;
   }
   new _ node ->data = data;
   new _ node -> next = stack -> top;
     stack -> top = new _ node;
}
```

```
int pop ( struct Stack* stack){
    if ( is_empty (stack) ){
        pf ("underflow");
        return -1;
 }
  struct Node* temp = stack -> top;
    int popped_data = temp -> data;
    stack -> top = temp -> next;
     free(temp);
   return popped_data;
 }

void display (struct Stack* stack){
      struct Node* current = stack -> top;
   while ( current ! =NULL){
      Pf ("%d", current -> data);
       current = current -> next;
   }
   Pf ("\n");
}

int main(){

     struct Stack stack;
       stack.top =NULL;

 int choice, dat;

do{
  Pf ("\n 1.Push \n 2.PoP \n 3. Display \n
       4. Exit \n");
Pf ("Enter your choice:");
Sf ("%d", &choice);

  switch (choice) {
   case 1:
        Pf("Enter data to push:");
        Sf("%d", &data);
        push (&stack, data);
        break;

   case 2 :
        Pf("Popped: %d\n", pop(&stack));
        break;

   case 3 :
        Pf ("stack:");
        display(&stack);
        break;

   case 4:
        Pf ("Exiting Program.\n");
        break;

   default:
        Pf ("Invalid choice");
  }
 }
 while (choice != 4);
 return0;
}
```



```
enter the operation:
1.push
2.pop
3.display
4.-1 to stop
enter operation
1
Enter the number:
5
enter operation
1
Enter the number:
6
enter operation
1
Enter the number:
7
enter operation
2
poped element is 7
enter operation
2
poped element is 6
enter operation
2
poped element is 5
enter operation
2
stack underflow
enter operation
```

Queue Implementation;

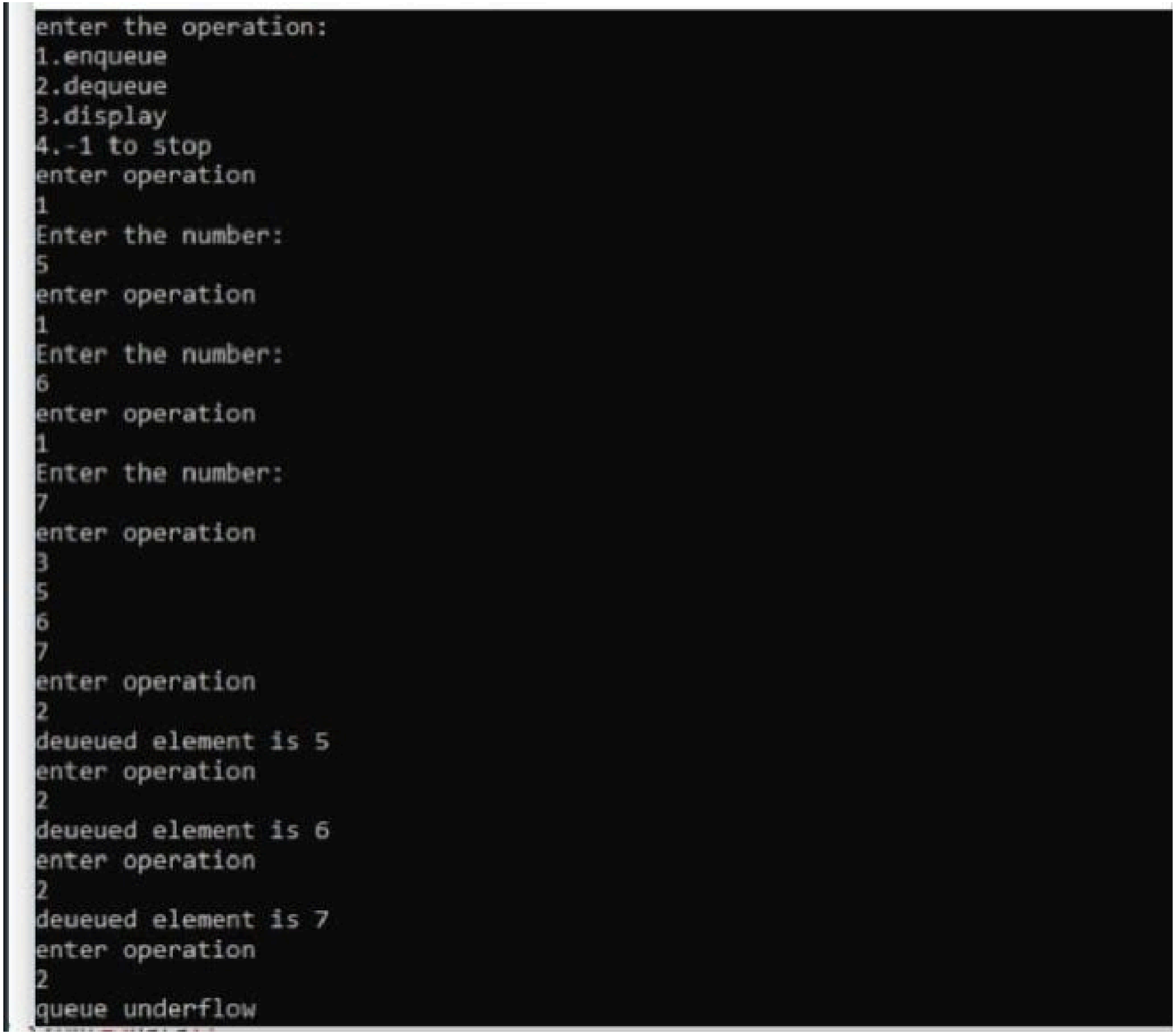
```
# include <stdio.h>
# include<stdlib.h>

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

  struct Queue {
          struct Node* front;
          struct Node* rear;
   };

  int is_empty (struct Queue* queue) {
         return queue -> front == NULL;
     }

 Void enqueue (struct Queue* queue, int data){
  struct Node* new_node = (struct Node*) malloc
                         (sizeof(struct Node));

   if (!new_node){
       pf ("Memory allocation error.\n");
     return;
   }
    new_node -> data = data;
    new_node -> next = NULL;
```

SL✓ - Queue Start

29/1/24

```
if (is-empty(queue)){
      queue->front = queue->rear = new-node;
  }
 else {
  queue->rear->next = new-node;
     queue->rear = new-node;
  }
 }

int deque (struct Queue *queue){

     if (is_empty(queue)) {
        Pf("Underflow");
        return -1;
     }

 struct Node * temp = queue->front;
 int dequed_data = temp->data;
 queue->front = temp->next;

 if (queue->front == NULL) {
        queue->reatr = NULL;
  }
   free(temp);
  return dequed_data;
}

Void display (struct Queue *queue){
struct Node * current = queue->front;
  while (current != NULL){
   Pf("%d", current->data);
     current = current->next;
  } Pf("\n");
}
```

```
int main(){
    struct Queue queue;
    queue.front = queue.rear = NULL;
    int choice, data;

  do{
  pf("\n 1. Enqueue \n 2. Dequeue \n 3. Display
      \n 4. Exit \n");
  pf("Enter your choice:");
  sf("%d", &choice);

  switch(choice){
    case 1:
            pf("Enter data to enqueue:");
            sf("%d", &data);
            enque(&queue,data);
            break;

    case 2:
          pf("Dequed :%d\n", dequeue(&queue));
          break;

    case 3:
        pf("Queue:");
        display(&queue);
        break;
    case 4:
            pf("Exit");
            break;
   deafault:
          pf("Invalid Choice");
  }}while(choice!=4);
    return0;
}
```



```
enter the operation:
1.enqueue
2.dequeue
3.display
4.-1 to stop
enter operation
1
Enter the number:
5
enter operation
1
Enter the number:
6
enter operation
1
Enter the number:
7
enter operation
3
5
6
7
enter operation
2
deueued element is 5
enter operation
2
deueued element is 6
enter operation
2
deueued element is 7
enter operation
2
queue underflow
```