Lab work – 8
Name- Yashwant Bhosale
MIS- 612303039
SY comp Div 1

Queue: Menu driven Program
Implement a queue of integers using an ADT list. Invoke all queue operations using a menu-driven program.
solution
Queue.h: Header file containing all structure declaration and function prototypes

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

typedef struct queue{
    node *head, *tail;
} queue;

void init(queue *q);
void enq(queue *q, int data);
int deq(queue *q);
int isEmpty(queue *q);
void print(queue *q);
void destroy(queue *q);
```

Queue.c: Contains all the function definitions for queue data structure

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include "queue.h"

/* Function to initialize queue */
void init(queue *q){
    if(!q) return;

    q→head = q→tail = NULL;
    return;
}
```

```
/* Function to insert an element to the queue */
void enq(queue *q, int data){
    if(!q) return;

    node *nn = (node *)malloc(sizeof(node));
    if(!nn) return; // Failed to allocate memory

    nn->data = data;
    nn->next = NULL;

    if(!q->head){
        q->head = q->tail = nn;
        return;
    }

    q->tail->next = nn;
    q->tail = nn;
    return;
}

/* Function to remove an element from the queue */
int deq(queue *q){
    if(!q) return INT_MIN;

    if(!q->head) return INT_MIN; // Queue is empty

    node *temp = q->head;
    int data = temp->data;

    q->head = q->head->next;
    free(temp);
    return data;
}

/* Function to check if queue is empty */
int isEmpty(queue *q){
    if(!q) return 1;

    return q->head == NULL;
}
```

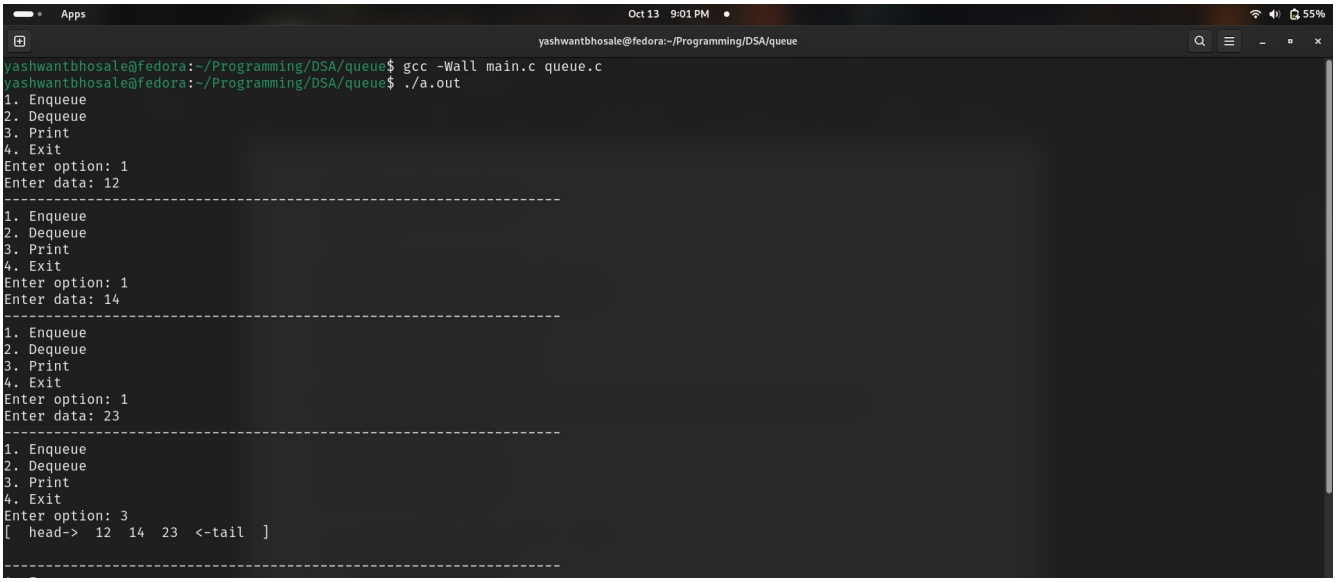
```
/* Function to free the queue */
void destroy(queue *q) {
    if(!q)  return;

    if(isEmpty(q)) return;

    node *p = q→head, *t = NULL;

    while(p→next) {
        t = p;
        free(t);
        p = p→next;
    }
    return;
}

/* Function to print the queue */
void print(queue *q){
    if(!q) return;
    node *p = q→head;
    printf("[  ");
    printf("head→  ");
    while(p){
        printf("%d  ", p→data);
        p = p→next;
    }
    printf("←tail  ");
    printf("]\n\n");
    return;
}
```

Main.c: Main flow of the program. Contains logic to handle menu and other details.

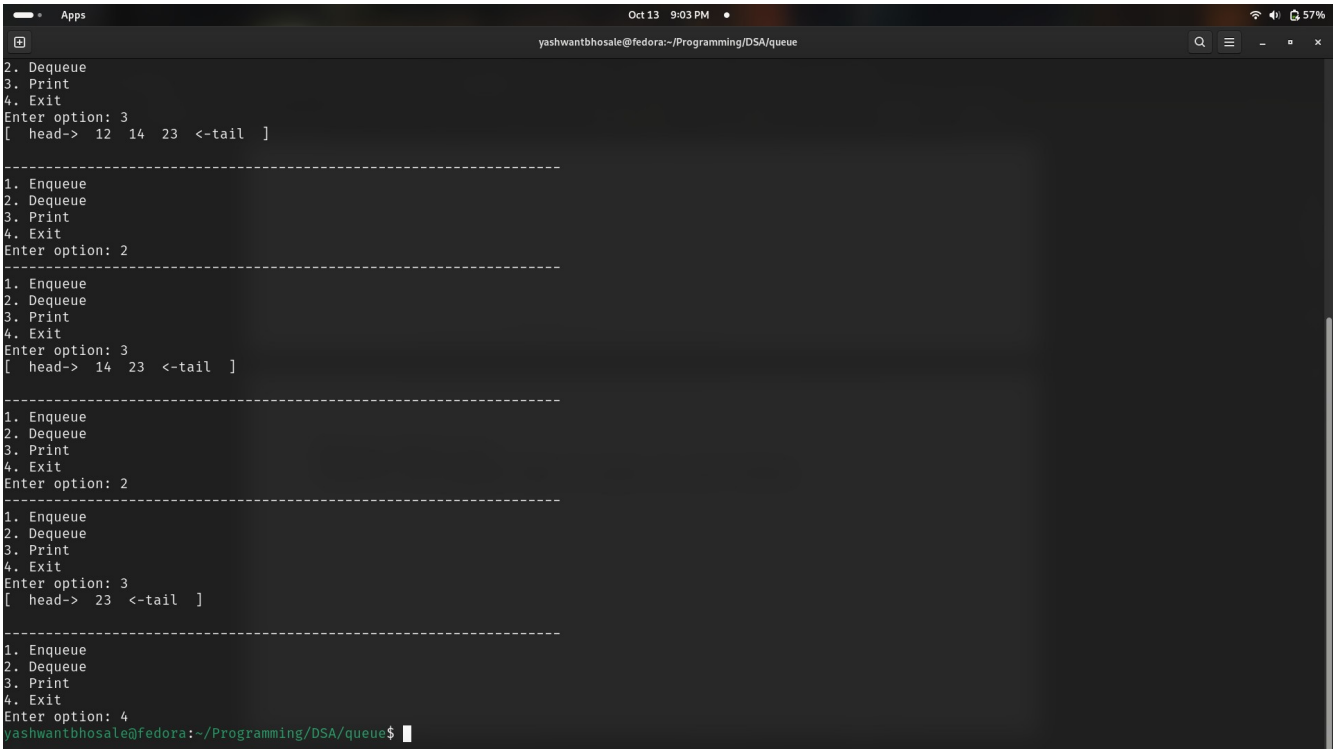
```
#include <stdio.h>
#include "queue.h"

void view_menu(){
    printf("1. Enqueue\n");
    printf("2. Dequeue\n");
    printf("3. Print\n");
    printf("4. Exit\n");
    return;
}

void evaluate_option(queue *q, int option){
    int data;
    switch(option){
        case 1:
            printf("Enter data: ");
            scanf("%d", &data);
            enq(q, data);
            break;
        case 2:
            deq(q);
            break;
        case 3:
            print(q);
            break;
        case 4:
            printf("Exiting...\n");
            break;
        default:
            printf("Invalid option\n");
    }


printf("————————————————————————————————————————————————————————————
————————\n");
    return;
}
```

```
int main() {
    queue q;
    init(&q);
    int option;
    while(1){
        view_menu();
        printf("Enter option: ");
        scanf("%d", &option);
        if(option == 4) break;
        evaluate_option(&q, option);
    }
    destroy(&q);
    return 0;
}
```

Output:
Enqueue operation:
Elements are inserted in the queue.



Dequeue Operation:
Elements are popped from the queue in LIFO format.



Lab Work 4: Stack

Question:

Implement a stack of integers using array. Invoke all stack functions using a menu driven program.

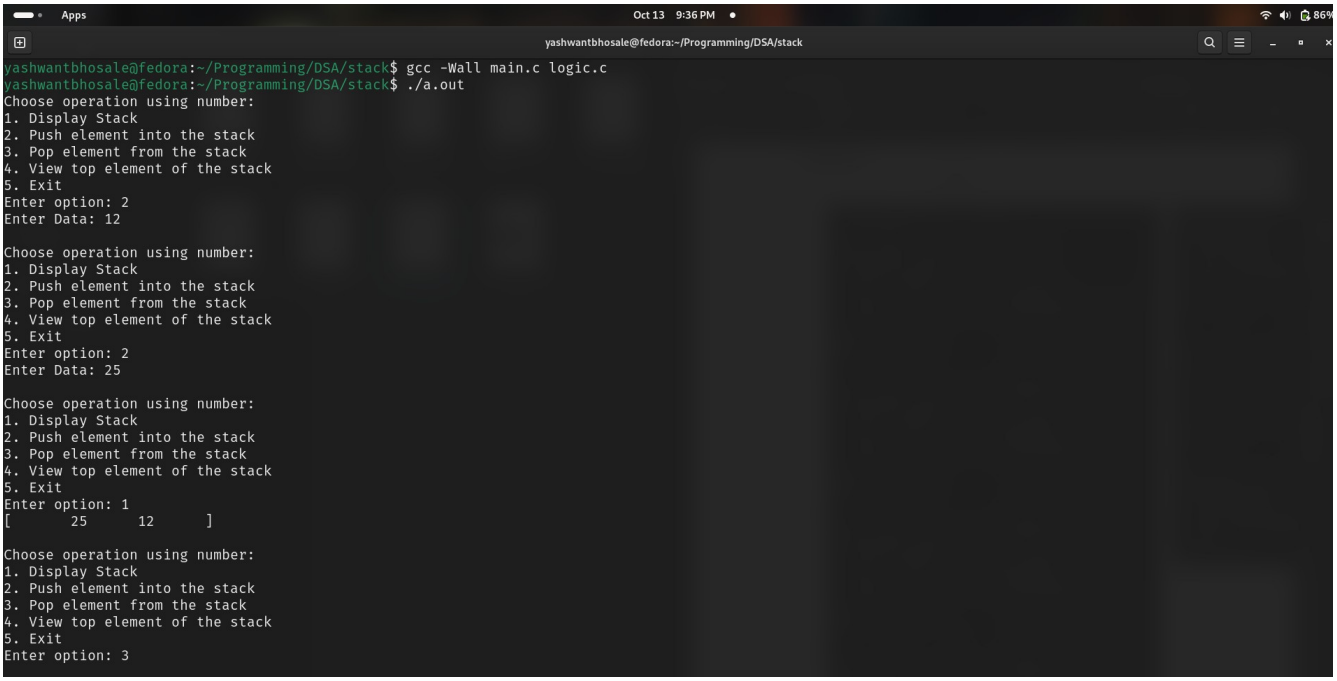
Code:

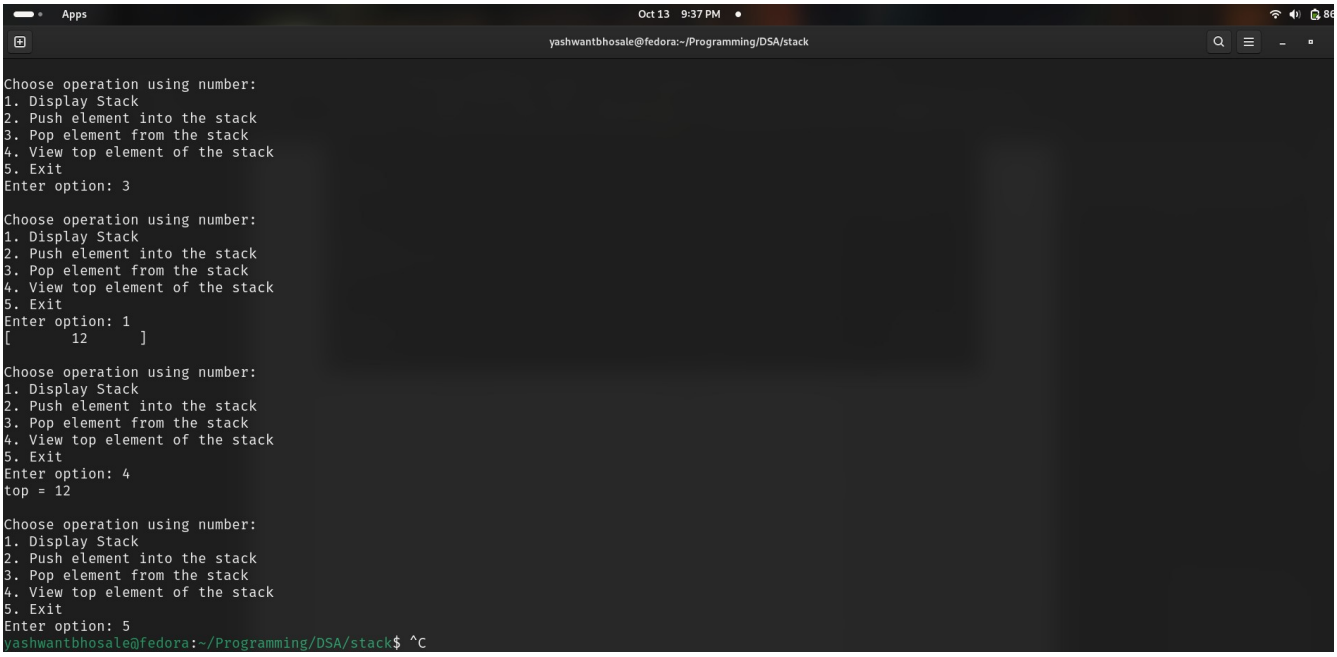
```
/* stack.h: Contains struct declarations and function prototypes */
typedef struct {
    int *arr;
    int size;
    int top;
} stack;

void init(stack *s, int size);
void push(stack *s, int data);
int pop(stack *s);
int peek(stack s);
void display(stack s);

/* logic.c: contains function definitions for functions */
#include "stack.h"

/* Function to initialize stack */
void init(stack *s, int size) {
    s → arr = (int *) malloc(size * sizeof(int));
    s → top = -1;
    s → size = size;
    return;
}

/* Function to push an element in the stack */
void push(stack *s, int data) {
    if(s→top ≥ s→size-1){
        s → arr = (int *) realloc(s→arr, (s→size+1) * sizeof(int));
    }
    s→top++;
    s→arr[s→top] = data;
    return;
}

/* Function to pop an element from the stack */
int pop(stack *s) {
    int element = s→arr[s→top];
    s→top--;
    return element;
}

/* Function to peek into the stack */
int peek(stack s) {
    return s.arr[s.top];
}
```

```c
/* Function to display the stack */
void display(stack s) {
    printf("[\t");
    for(int i = s.top; i ⩾ 0; i--) {
        printf("%d\t", s.arr[i]);
    }
    printf("]\n");
    return;
}

/* main.c: menu driven flow for the program */
#include <stdio.h>
#include <stdlib.h>
#include "stack.h"

void display_menu() {
    printf("Choose operation using number:\n");
    printf("1. Display Stack\n");
    printf("2. Push element into the stack\n");
    printf("3. Pop element from the stack\n");
    printf("4. View top element of the stack\n");
    printf("5. Exit\n");
    return;
}

void read_option(int option, stack *s) {
    switch (option){
    case 1:{
        display(*s);
        break;
    }
    case 2: {
        int data;
        printf("Enter Data: ");
        scanf("%d", &data);
        push(s, data);
        break;
    }
    case 3: {
        pop(s);
        break;
    }
    case 4: {
        printf("top = %d\n", peek(*s));
        break;
    }
    default:
        printf("Invalid Option\n");
        break;
    }
    return;
}
```

```
int main() {
    int option = 0;
    stack s;
    init(&s, 3);
    while(1) {
        display_menu();
        printf("Enter option: ");
        scanf("%d", &option);
        if(option == 5)
            break;
        read_option(option, &s);
        printf("\n");
    }

    return 0;
}
```

Output:


Apps
Oct 13 9:36 PM
yashwantbhosale@fedora:~/Programming/DSA/stack
yashwantbhosale@fedora:~/Programming/DSA/stack$ gcc -Wall main.c logic.c
yashwantbhosale@fedora:~/Programming/DSA/stack$ ./a.out
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 2
Enter Data: 12
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 2
Enter Data: 25
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 1
[25 12]
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 3


Apps
Oct 13 9:37 PM
yashwantbhosale@fedora:~/Programming/DSA/stack
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 3
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 1
[12]
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 4
top = 12
Choose operation using number:
1. Display Stack
2. Push element into the stack
3. Pop element from the stack
4. View top element of the stack
5. Exit
Enter option: 5
yashwantbhosale@fedora:~/Programming/DSA/stack$ ^C