

14/01/2022

Data Structures

Term Test 2

Solutions:

1) Code:

```
#include <stdio.h>
#define SIZE 5
void insert();
void delete();
void display();
int queue[SIZE], rear = -1, front = -1, item;
main()
{
    int ch;
    do
    {
        printf("\n 1. Insert 2. Delete 3. Insert Display 4. Exit ");
        printf("\n Enter your choice ");
        scanf("%d", &ch);
        switch(ch)
        {
            case 1:
                insert();
                break;
            case 2:
                delete();
                break;
            case 3:
                display();
                break;
        }
    }
}
```

Case 4 :

```
exit ( ) ;
```

default :

```
printf ( " \n Invalid choice . Please try again . \n " ) ;
```

```
}
```

```
} while ( 1 ) ;
```

```
{
```

```
void insert ( )
```

```
{
```

```
if ( ( front == 0 && rear == SIZE - 1 ) || ( front == rear + 1 ) )
```

```
printf ( " \n Queue is full " ) ;
```

```
else
```

```
{
```

```
printf ( " \n Enter ITEM " ) ;
```

```
scanf ( " %d " , & item ) ;
```

```
if ( rear == - 1 )
```

```
{
```

```
rear = 0 ;
```

```
front = 0 ;
```

```
}
```

```
else if ( rear == SIZE - 1 )
```

```
rear = 0 ;
```

```
else {
```

```
rear ++ ;
```

```
queue [ rear ] = item ;
```

```
printf ( " \n Item inserted : %d " , item ) ;
```

```
}
```

```
}
```

```
}
```

```
void delete()
```

```
{
```

```
    if (front == -1)
```

```
        printf("In Queue is empty\n");
```

```
    else
```

```
    {
```

```
        item = queue[front];
```

```
        if (front == rear)
```

```
        {
```

```
            front = -1;
```

```
            rear = -1;
```

```
        }
```

```
        else if (front == SIZE - 1)
```

```
            front = 0;
```

```
        else
```

```
            front ++;
```

```
        printf("ITEM deleted: %d", item);
```

```
    }
```

```
}
```

```
void display()
```

```
{
```

```
    int i;
```

```
    if ((front == -1) || (front == rear + 1))
```

```
        printf("In Queue is empty\n");
```

```
    else {
```

```
        printf("\n\n");
```

```
        for (i = front, i <= rear; i++) {
```

```
            printf(" %d", queue[i]);
```

```
        }
```

```
    }
```

```
}
```


3> Code :

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

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

```
typedef struct stack {
```

```
    int data;
```

```
    struct stack *next;
```

```
} stack;
```

```
stack *push (stack *top) {
```

```
    int val;
```

```
    stack *ptr;
```

```
    printf("Enter value to be added ");
```

```
    scanf("%d", &val);
```

```
    ptr = (stack *) malloc (sizeof (stack));
```

```
    ptr->data = val;
```

```
    if (top == NULL) {
```

```
        top = ptr;
```

```
        ptr->next = NULL;
```

```
    }
```

```
    else {
```

```
        ptr->next = top;
```

```
        top = ptr;
```

```
    }
```

```
    return top;
```

```
}
```

```
Stack * pop (Stack * top) {
```

```
    Stack * temp;
```

```
    temp = top;
```

```
    if (temp == NULL) {
```

```
        printf ("Stack underflow" );
```

```
    }
```

```
    else {
```

```
        top = top → next ;
```

```
        printf ("The element popped is '%d'", temp → data);
```

```
        free (temp);
```

```
    }
```

```
    return top;
```

```
}
```

```
Stack * display (Stack * top) {
```

```
    Stack * temp;
```

```
    temp = top;
```

```
    if (top == NULL) {
```

```
        printf ("The stack is empty" );
```

```
    }
```

```
    else {
```

```
        while (temp → next != NULL) {
```

```
            printf ("%d", temp → data);
```

```
            temp = temp → next;
```

```
        }
```

```
        printf ("%d", temp → data);
```

```
    }
```

```
    return top;
```

```
}
```

```

void peek (stack *top) {
    if (top == NULL) {
        printf("The stack is empty" );
    }
    else {
        printf(" %d ", top -> data);
    }
}

```

```

void main() {
    int choice;
    stack *top;
    top = NULL;
    do {
        printf("In menu In 1 to push In 2 to pop In 3 to display In  

            4 to peek In 5 to exist In Enter your choice:");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                top = push(top);
                break;
            case 2:
                top = pop(top);
                break;
            case 3:
                top = display (top);
                break;

```


case 4 :

peek (top);

break;

case 5 :

break;

default :

printf("The choice entered is invalid\n");

}

} while (choice != 5);

}

2) Code :

void insertBefore() {

int val, pos;

printf("Enter before which element you want to enter");

scanf("%d", &pos)

2> Code:

```
void insert_before ()
```

```
{
```

```
    int num;
```

```
    Node *newnode, *temp;
```

```
    newnode = (Node *) malloc (sizeof (Node));
```

```
    printf ("Before which number you want to insert");
```

```
    scanf ("%d", &num);
```

```
    printf ("Enter element: ");
```

```
    scanf ("%d", &newnode->data);
```

```
    if (num == head->data)
```

```
    {
```

```
        newnode->next = head;
```

```
        head = newnode;
```

```
    }
```

```
    else
```

```
    {
```

```
        Node *t;
```

```
        temp = head;
```

```
        t = head;
```

```
        temp = temp->next;
```

```
        while (temp->data != num)
```

```
        {
```

```
            temp = temp->next;
```

```
            t = t->next;
```

```
        }
```

```
        newnode->next = temp;
```

```
        t->next = newnode;
```

```
    }
```

```
}
```

FOR EDUCATIONAL USE