DATA STRUCTURES AND ALGORITHMS

CYCLESHEET-1

NAME: ANUBHAV JAIN
REG.NO: 22BIT0210
FACULTY: PROF VIJAYAN.E

Data Structures and Algorithms

Cyclesheet-1

Name: Anubhav Jain

Reg.No:22BIT0210

1.Write a C program to implement stack using array

Source code:-

```
//20-08-2023
//Anubhav Jain
//22BIT0210

#include<stdio.h>
struct stack{
int arr[5];
int top;
}st;

void stack_push() {
int item;
if(st.top==4) {
printf("Stack Overflow\n");
}
else{
printf("Enter the item: ");
scanf("%d", &item);
printf("**************\n");
st.top++;
st.arr[st.top]=item;
}
}
void stack_pop()
{
    if(st.top==-1)
    {
        printf("Stack underflow");
```

```
    }
    else{
        st.top--;
    }
}
void stack_display()
{
    if(st.top==-1)
    {
        printf("Stack underflow\n");
    }
    else
    {
        int temp=st.top;
        while(temp>=0)
        {   printf("Stack:");
            printf("%d",st.arr[temp]);
            printf("\n");
            temp--;
        }
    }
}
int main()
{
    st.top=-1;
    int choice;
    printf("\nANUBHAV JAIN 22BIT0210\n");
    do
    {

printf("1.Insertion\n2.Deletion\n3.Display\n4.Exit\n");
        printf("Enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
        case 1:
            stack_push();
            break;
        case 2:
            stack_pop();
            break;
        case 3:
            stack_display();
            break;
        case 4:
```

```
                exit(0);
                break;
            }
        }
        while(choice<4);
}
```

Output:-

```
C:\Users\rajiv\OneDrive\Desk

ANUBHAV JAIN 22BIT0210
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
3
Stack underflow
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
1
Enter the item: 10
**************
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
1
Enter the item: 11
**************
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
1
Enter the item: 12
**************
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
1
Enter the item: 13
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the choice
1
Enter the item: 14
**************
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
1
Stack Overflow
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
3
Stack:14
Stack:13
Stack:12
Stack:11
Stack:10
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
2
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
2
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
3
Stack:12
Stack:11
Stack:10
1.Insertion
2.Deletion
3.Display
4.Exit
Enter the choice
4

Process returned 0 (0x0)   execution time : 78.297 s
Press any key to continue.
```

2. write a C program to implement in-fix to post-fix expression

Source code:-

```
#include <stdio.h>
//ANUBHAV JAIN
//2BIT0210
char stack[100];
int top = -1;

void push(char x) {
    stack[++top] = x;
}

char pop() {
    if (top == -1) {
        return -1;
    } else {
        return stack[top--];
    }
}

int precedence(char x) {
    if (x == '(') {
        return 0;
    }
    if (x == '+' || x == '-') {
        return 1;
    }
    if (x == '*' || x == '/') {
        return 2;
    }
    return 0;
}

int main() {
    char exp[100];
    char *e, x;
    printf("ANUBHAV JAIN 22BIT0210\n");
    printf("Enter the Expression: ");
    scanf("%s", exp);
    printf("\n");
    e = exp;
```

```
    while (*e != '\0') {
        if (isalnum(*e)) {
            printf("%c", *e);
        } else if (*e == '(') {
            push(*e);
        } else if (*e == ')') {
            while ((x = pop()) != '(') {
                printf("%c", x);
            }
        } else {
            while (top != -1 && precedence(stack[top])
>= precedence(*e)) {
                printf("%c", pop());
            }
            push(*e);
        }
        e++;
    }
    while (top != -1) {
        printf("%c", pop());
    }
    return 0;
}
```

Output:-

```
C:\Users\rajiv\OneDrive\Desk

ANUBHAV JAIN 22BIT0210
Enter the Expression: (A+B)*(C-D)

AB+CD-*
Process returned 0 (0x0)   execution time : 52.158 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

ANUBHAV JAIN 22BIT0210
Enter the Expression: (((A+B)*(C-D)/E)/F*G)

AB+CD-*E/F/G*
Process returned 0 (0x0)   execution time : 31.625 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

ANUBHAV JAIN 22BIT0210
Enter the Expression: A+B*C/D/E*F

ABC*D/E/F*+
Process returned 0 (0x0)   execution time : 22.797 s
Press any key to continue.
```

3. Write a C program to implement in-fix to pre-fix expression

Source code:-

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

//ANIBHAV JAIN
//22BIT0210

struct Stack {
    int top;
    int capacity;
    char* array;
};

struct Stack* createStack( int capacity)
{
    struct Stack* stack = (struct
Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = (char*)malloc(stack->capacity *
sizeof(char));
    return stack;
}

int isEmpty(struct Stack* stack)
{
    return stack->top == -1;
}

char peek(struct Stack* stack)
{
    return stack->array[stack->top];
}

char pop(struct Stack* stack)
{
    if (!isEmpty(stack))
        return stack->array[stack->top--];
    return '$';
}
```

```
void push(struct Stack* stack, char op)
{
    stack->array[++stack->top] = op;
}

int isOperand(char ch)
{
    return (ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch
<= 'Z');
}

int isOperator(char ch)
{
    return ch == '+' || ch == '-' || ch == '*' || ch ==
'/' || ch == '^';
}

int precedence(char ch)
{
    if (ch == '^')
        return 3;
    else if (ch == '*' || ch == '/')
        return 2;
    else if (ch == '+' || ch == '-')
        return 1;
    else
        return -1;
}

void infixToPrefix(char* infix, char* prefix)
{
    struct Stack* stack = createStack(strlen(infix));
    int i, j = 0;
    for (i = strlen(infix) - 1; i >= 0; i--) {
        if (isOperand(infix[i])) {
            prefix[j++] = infix[i];
        }
        else if (infix[i] == ')') {
            push(stack, infix[i]);
        }
        else if (infix[i] == '(') {
            while (!isEmpty(stack) && peek(stack) !=
')') {
                prefix[j++] = pop(stack);
```

```
            }
            if (!isEmpty(stack) && peek(stack) != ')')
{
                printf("Invalid expression");
                return;
            }
            else {
                pop(stack);
            }
        }
        else if (isOperator(infix[i])) {
            while (!isEmpty(stack) &&
precedence(infix[i]) < precedence(peek(stack))) {
                prefix[j++] = pop(stack);
            }
            push(stack, infix[i]);
        }
    }
    while (!isEmpty(stack)) {
        prefix[j++] = pop(stack);
    }
    prefix[j] = '\0';
    strrev(prefix);
}

int main()
{
    char infix[100], prefix[100];
    printf("Enter an infix expression: ");
    gets(infix);
    infixToPrefix(infix, prefix);
    printf("Prefix expression: %s", prefix);
    return 0;
}
```

OUTPUT:-

```
C:\Users\rajiv\OneDrive\Desk

Enter an infix expression: (A+B)*(C-D)
Prefix expression: *+AB-CD
Process returned 0 (0x0)   execution time : 17.685 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter an infix expression: (((A+B)*(C-D)/E)/F*G)
Prefix expression: *//*+AB-CDEFG
Process returned 0 (0x0)   execution time : 133.206 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter an infix expression: A+B*C/D/E*F
Prefix expression: +A*//*BCDEF
Process returned 0 (0x0)   execution time : 18.375 s
Press any key to continue.
```

4. Write a C program to implement evaluation of post-fix

SOURCE CODE:

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

//ANUBHAV JAIN
//22BIT0210
char stack[100];
int top = -1;

void push(int x) {
    if (top >= 100 - 1) {
        printf("Stack overflow\n");
    } else {
        stack[++top] = x;
    }
}

int pop() {
    if (top == -1) {
        printf("Stack underflow\n");
        return -1;
    } else {
        return stack[top--];
    }
}

int evaluatePostfix(char* postfix) {
    int i = 0;
    for (i = 0; postfix[i] != '\0'; i++) {
        if (postfix[i] >= '0' && postfix[i] <= '9') {
            push(postfix[i] - '0');
        } else {
            int operand2 = pop();
            int operand1 = pop();
            int result;

            switch (postfix[i]) {
                case '+':
                    result = operand1 + operand2;
                    break;
```

```
                case '-':
                    result = operand1 - operand2;
                    break;
                case '*':
                    result = operand1 * operand2;
                    break;
                case '/':
                    result = operand1 / operand2;
                    break;
            }

            push(result);
        }
    }

    return pop();
}

int main() {
    char postfix[100];
    printf("Enter a postfix expression: ");
    scanf("%s", postfix);

    int result = evaluatePostfix(postfix);
    printf("Result: %d\n", result);

    return 0;
}
```

OUTPUT:-

```
C:\Users\rajiv\OneDrive\Desk

Enter a postfix expression: 23+64-*
Result: 10

Process returned 0 (0x0)   execution time : 3.470 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter a postfix expression: 234*+
Result: 14

Process returned 0 (0x0)   execution time : 6.875 s
Press any key to continue.
```

5. Write a C program to implement evaluation of pre-fix

SOURCE CODE:-

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

//ANUBHAV JAIN
//22BIT0210

int stack[100];
int top = -1;

void push(int x) {
    if (top >= 100- 1) {
        printf("Stack overflow\n");
    } else {
        stack[++top] = x;
    }
}

int pop() {
    if (top == -1) {
        printf("Stack underflow\n");
        return -1;
    } else {
        return stack[top--];
    }
}

int evaluatePrefix(char* prefix) {
    int i, len = strlen(prefix);

    for (i = len - 1; i >= 0; i--) {
        if (prefix[i] >= '0' && prefix[i] <= '9') {
            push(prefix[i] - '0');
        } else {
            int operand1 = pop();
            int operand2 = pop();
            int result;

            switch (prefix[i]) {
                case '+':
```

```
                    result = operand1 + operand2;
                    break;
                case '-':
                    result = operand1 - operand2;
                    break;
                case '*':
                    result = operand1 * operand2;
                    break;
                case '/':
                    result = operand1 / operand2;
                    break;
            }

            push(result);
        }
    }

    return pop();
}

int main() {
    char prefix[100];
    printf("Enter a prefix expression: ");
    scanf("%s", prefix);

    int result = evaluatePrefix(prefix);
    printf("Result: %d\n", result);

    return 0;
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Enter a prefix expression: +2*43
Result: 14

Process returned 0 (0x0)   execution time : 6.112 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter a prefix expression: *+23-64
Result: 10

Process returned 0 (0x0)   execution time : 15.473 s
Press any key to continue.
```

6. Write a C program to implement towers of Hanoi problem

Source Code:-

```
#include <stdio.h>
//Anubhav Jain
//22BIT0210
void towers(int, char, char, char);

int main()
{
    int num;

    printf("Enter the number of disks : ");
    scanf("%d", &num);
    printf("The sequence of moves involved in the Tower
of Hanoi are :\n");
    towers(num, 'A', 'C', 'B');
    return 0;
}
void towers(int num, char frompeg, char topeg, char
auxpeg)
{
    if (num == 1)
    {
        printf("\n Move disk 1 from tower %c to tower
%c", frompeg, topeg);
        return;
    }

    towers(num - 1, frompeg, auxpeg, topeg);
    printf("\n Move disk %d from tower %c to tower %c",
num, frompeg, topeg);
    towers(num - 1, auxpeg, topeg, frompeg);
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of disks : 3
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from tower A to tower C
 Move disk 2 from tower A to tower B
 Move disk 1 from tower C to tower B
 Move disk 3 from tower A to tower C
 Move disk 1 from tower B to tower A
 Move disk 2 from tower B to tower C
 Move disk 1 from tower A to tower C
Process returned 0 (0x0)   execution time : 2.073 s
Press any key to continue.
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the number of disks : 4
The sequence of moves involved in the Tower of Hanoi are :

 Move disk 1 from tower A to tower B
 Move disk 2 from tower A to tower C
 Move disk 1 from tower B to tower C
 Move disk 3 from tower A to tower B
 Move disk 1 from tower C to tower A
 Move disk 2 from tower C to tower B
 Move disk 1 from tower A to tower B
 Move disk 4 from tower A to tower C
 Move disk 1 from tower B to tower C
 Move disk 2 from tower B to tower A
 Move disk 1 from tower C to tower A
 Move disk 3 from tower B to tower C
 Move disk 1 from tower A to tower B
 Move disk 2 from tower A to tower C
 Move disk 1 from tower B to tower C
Process returned 0 (0x0)   execution time : 0.777 s
Press any key to continue.
```

7. Write a C program to implement Linear Queue using array

SOURCE CODE:

```
#include <stdio.h>
#include <stdlib.h>
//ANUBHAV JAIN
//22BIT0210

struct Queue {
    int items[5];
    int front;
    int rear;
};

struct Queue q;

void enqueue(int value) {
    if (q.rear == 5 - 1) {
        printf("Queue is full\n");
    } else {
        if (q.front == -1) {
            q.front = 0;
        }
        q.rear++;
        q.items[q.rear] = value;
        printf("Inserted: %d\n", value);
    }
}

void dequeue() {
    if (q.front == -1) {
        printf("Queue is empty\n");
    } else {
        printf("Deleted: %d\n", q.items[q.front]);
        q.front++;
        if (q.front > q.rear) {
            q.front = q.rear = -1;
        }
    }
}

void display() {
    if (q.front == -1) {
        printf("Queue is empty\n");
```

```
    } else {
        printf("Linear Queue: ");
        for (int i = q.front; i <= q.rear; i++) {
            printf("%d ", q.items[i]);
        }
        printf("\n");
    }
}

int main() {
    q.front = -1;
    q.rear = -1;

    int choice, value;

    do {
        printf("\nQueue Operations:\n");
        printf("1.Insertion\n");
        printf("2.Deletion\n");
        printf("3.Display\n");
        printf("4.Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program\n");
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    } while (1);

    return 0;
```

```
}
```

Output:

```
C:\Users\rajiv\OneDrive\Desk

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 3
Queue is empty

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 10
Inserted: 10

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 20
Inserted: 20

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 30
Inserted: 30

Queue Operations:
1.Insertion
```

```
C:\Users\rajiv\OneDrive\Desk

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 40
Inserted: 40

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 50
Inserted: 50

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 60
Queue is full

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 3
Linear Queue: 10 20 30 40 50

Queue Operations:
1.Insertion
```

```
C:\Users\rajiv\OneDrive\Desk

Linear Queue: 10 20 30 40 50

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 2
Deleted: 10

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 2
Deleted: 20

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 3
Linear Queue: 30 40 50

Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 4
Exiting program

Process returned 0 (0x0)   execution time : 375.795 s
Press any key to continue.
```

8. Write a C program to implement circular queue

Source Code:

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

//ANUBHAV JAIN
//22BIT0210

struct Cqueue {
    int a[5];
    int front;
    int rear;
}cq;


void cqueue_insertion_rear()
{   int item;
    if (cq.front==(cq.rear+1)%5)
        {
        printf("Queue is full\n");
    }
    else
    {
        printf("Enter the item");
        scanf("%d",&item);
        if(cq.rear==-1)
        {
            cq.rear++;
            cq.front++;
            cq.a[cq.rear]=item;
        }
        else
        { cq.rear=((cq.rear+1)%5);
          cq.a[cq.rear]=item;

        }
    }
}

void queue_delete() {
    if (cq.rear == -1) {
        printf("Queue is empty\n");
```

```
    }
    else
    {
        if (cq.front == cq.rear)
        {
            cq.front = cq.rear = -1;
        }
        else
        {
            cq.front = ((cq.front+1)%5);
        }
    }
}

void display()
{ int temp;
  if(cq.rear==-1)
    printf("queue empty");
  else
  {
      temp=cq.front;
      while(temp!=cq.rear)
      {
          printf("%d",cq.a[temp]);
          temp=(temp+1)%5;
      }
      printf("%d",cq.a[temp]);
  }

}

int main()
{   int choice;
    cq.front = -1;
    cq.rear = -1;

    do {
        printf("\nCircular Queue Operations:\n");
        printf("1.Insertion\n");
        printf("2.Deletion\n");
        printf("3.Display\n");
        printf("4.Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
```

```
        switch (choice) {
            case 1:
                cqueue_insertion_rear();
                break;
            case 2:
                queue_delete();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program\n");
                exit(0);
            default:
                printf("Invalid choice\n");
        }
    } while (1);

    return 0;
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 3
queue empty
Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter the item:10

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter the item:20

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter the item:30

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter the item:40
```

```
C:\Users\rajiv\OneDrive\Desk

Enter the item:40

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Enter the item:50

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 1
Queue is full

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 3
10 20 30 40 50
Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 2

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 2

Circular Queue Operations:
```

```
C:\Users\rajiv\OneDrive\Desk

2.Deletion
3.Display
4.Quit
Enter your choice: 3
10 20 30 40 50
Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 2

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 2

Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 3
30 40 50
Circular Queue Operations:
1.Insertion
2.Deletion
3.Display
4.Quit
Enter your choice: 4
Exiting program

Process returned 0 (0x0)   execution time : 223.626 s
Press any key to continue.
```

9. Write a C program to implement Ascending priority queue

Source Code:

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

//Anubhav Jain
//22BIT0210

struct PriorityQueue {
    int elements[5];
    int rear;
};

struct PriorityQueue pq;

void enqueue(int value) {
    if (pq.rear == 5 - 1) {
        printf("Priority Queue is full\n");
        return;
    }

    int i = pq.rear;
    while (i >= 0 && pq.elements[i] > value) {
        pq.elements[i + 1] = pq.elements[i];
        i--;
    }

    pq.elements[i + 1] = value;
    pq.rear++;
}

void dequeue() {
    if (pq.rear == -1) {
        printf("Priority Queue is empty\n");
    } else {
        printf("Deleted: %d\n", pq.elements[0]);

        for (int i = 0; i < pq.rear; i++) {
            pq.elements[i] = pq.elements[i + 1];
        }

        pq.rear--;
```

```
    }
}

void display() {
    if (pq.rear == -1) {
        printf("Priority Queue is empty\n");
    } else {
        printf("Priority Queue elements:\n");
        for (int i = 0; i <= pq.rear; i++) {
            printf("Value: %d\n", pq.elements[i]);
        }
    }
}

int main() {
    pq.rear = -1;

    int choice, value;

    do {
        printf("\nPriority Queue Operations:\n");
        printf("1.Enqueue\n");
        printf("2.Dequeue\n");
        printf("3.Display\n");
        printf("4.Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program\n");
                exit(0);
            default:
                printf("Invalid choice\n");
```

```
        }
    } while (1);

    return 0;
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 3
Priority Queue is empty

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 76

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 45

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 67

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 34
```

```
C:\Users\rajiv\OneDrive\Desk

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 89

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 33
Priority Queue is full

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 3
Priority Queue elements:
Value: 34
Value: 45
Value: 67
Value: 76
Value: 89

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 2
Deleted: 34

Priority Queue Operations:
```

```
C:\Users\rajiv\OneDrive\Desk

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 2
Deleted: 45

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 2
Deleted: 45

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 3
Priority Queue elements:
Value: 67
Value: 76
Value: 89

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 4
Exiting program

Process returned 0 (0x0)
Press any key to continue.
```

10. Write a C program to implement Descending priority queue

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
//Anubhav Jain
//22BIT0210

struct PriorityQueue {
    int elements[5];
    int rear;
};

struct PriorityQueue pq;

void enqueue(int value) {
    if (pq.rear == 5 - 1) {
        printf("Priority Queue is full\n");
        return;
    }

    int i = pq.rear;
    while (i >= 0 && pq.elements[i] < value) {
        pq.elements[i + 1] = pq.elements[i];
        i--;
    }

    pq.elements[i + 1] = value;
    pq.rear++;
}

void dequeue() {
    if (pq.rear == -1) {
        printf("Priority Queue is empty\n");
    } else {
        printf("Deleted: %d\n", pq.elements[0]);

        for (int i = 0; i < pq.rear; i++) {
            pq.elements[i] = pq.elements[i + 1];
        }

        pq.rear--;
    }
```

```
}

void display() {
    if (pq.rear == -1) {
        printf("Priority Queue is empty\n");
    } else {
        printf("Priority Queue elements:\n");
        for (int i = 0; i <= pq.rear; i++) {
            printf("Value: %d\n", pq.elements[i]);
        }
    }
}

int main() {
    pq.rear = -1;

    int choice, value;

    do {
        printf("\nPriority Queue Operations:\n");
        printf("1.Enqueue\n");
        printf("2.Dequeue\n");
        printf("3.Display\n");
        printf("4.Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program\n");
                exit(0);
            default:
                printf("Invalid choice\n");
        }
```

```
    } while (1);

    return 0;
}
```

Output:

```
C:\Users\rajiv\OneDrive\Desk

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 3
Priority Queue is empty

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 24

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 67

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 45

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 54
```

```
C:\Users\rajiv\OneDrive\Desk

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 88

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 1
Enter value to enqueue: 33
Priority Queue is full

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 3
Priority Queue elements:
Value: 88
Value: 67
Value: 54
Value: 45
Value: 24

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 2
Deleted: 88

Priority Queue Operations:
1.Enqueue
```

```
C:\Users\rajiv\OneDrive\Desk

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 2
Deleted: 88

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 2
Deleted: 67

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 3
Priority Queue elements:
Value: 54
Value: 45
Value: 24

Priority Queue Operations:
1.Enqueue
2.Dequeue
3.Display
4.Quit
Enter your choice: 4
Exiting program

Process returned 0 (0x0)   executic
Press any key to continue.
```

CHALLENGING EXPERIMENT

Students of a Programming class arrive to submit assignments. Their register numbers are stored in a LIFO list in the order in which the assignments are submitted. Write a program using array to display the register number of

the ten students who submitted first.

Register number of the ten students who submitted first will be at the bottom of the LIFO list. Hence pop out the required number of elements from the top so as to retrieve and display the first 10 students.

Source Code:

```
#include<stdio.h>
#include<string.h>
//ANUBHAV JAIN
//22BIT0210
struct Stack {
    char data[70][70];
    int top;
} st;

void Push(char ele[]) {
    if (st.top + 1 > 70) {
        printf("\nStack Overflow\n");
    } else {
        st.top++;
        strcpy(st.data[st.top], ele);
    }
}

void Pop() {
    if (st.top == -1) {
        printf("Stack Underflow");
    } else {
        printf("Deleted Register No. = %s\n",
st.data[st.top]);
        st.top--;
    }
}

void Disp() {
    int i;
```

```
    for (i = st.top; i >= 0; i--) {
        printf("\n%s", st.data[i]);
    }
    printf("\n");
}

int main() {
    int n, i, j;
    char val[70];
    st.top = -1;
    printf("Enter number of students: ");
    scanf("%d", &n);
    printf("Now, enter the reg. nos\n");
    for (i = 0; i < n; i++) {
        int f = 1;
        scanf("%s", val);
        if (st.top != -1) {
            int t = st.top;
            for (j = 0; j <= t; j++) {
                if (strcmp(val, st.data[j]) == 0) {
                    f = 0;
                    printf("\nNumber already
Exists\n");
                    i--;
                    break;
                }
            }
            if (f == 0) continue;
            Push(val);
        } else {
            Push(val);
        }
    }
    int m = 10;
    int t = st.top;
    for (i = st.top; i >= m; i--) {
        Pop();
    }
    printf("\nNow Printing the first %d Numbers:\n",
m);
    Disp();
    return 0;
}
```

OUTPUT:

```
C:\Users\rajiv\OneDrive\Desk

Enter number of students: 13
Now, enter the reg. nos
22BIT0210
22BIT0193
22BIT0187
22BIT0924
22BIT0234
22BIT0724
22BIT0232
22BIT0324
22BIT0353
22BIT0687
22BIT777
22BIT0243
22BIT0222
Deleted Register No. = 22BIT0222
Deleted Register No. = 22BIT0243
Deleted Register No. = 22BIT777

Now Printing the first 10 Numbers:

22BIT0687
22BIT0353
22BIT0324
22BIT0232
22BIT0724
22BIT0234
22BIT0924
22BIT0187
22BIT0193
22BIT0210

Process returned 0 (0x0)   execution time : 55.034 s
Press any key to continue.
```