IMPLEMENTATION OF QUEUE USING ARRAY OPERATIONS

```
#include<stdio.h>

#include<stdlib.h>

#define maxsize 5

void insert();

void delete();

void display();

int front = -1, rear = -1;

int queue[maxsize];

void main ()

{

	int choice;

	while(choice != 4)

	{

		printf("\n************************Main Menu****************************\n");

		printf("\n==================================================================\n");

		printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

		printf("\nEnter your choice ?");

		scanf("%d",&choice);

		switch(choice)

		{

			case 1:

			insert();

			break;

			case 2:
```

```
                delete();
                break;
                case 3:
                display();
                break;
                case 4:
                exit(0);
                break;
                default:
                printf("\nEnter valid choice??\n");
            }
        }
}
void insert()
{
        int item;
        printf("\nEnter the element\n");
        scanf("\n%d",&item);
        if(rear == maxsize-1)
        {
            printf("\nOVERFLOW\n");
            return;
        }
        if(front == -1 && rear == -1)
        {
            front = 0;
```

```
            rear = 0;
        }
        else
        {
            rear = rear+1;
        }
        queue[rear] = item;
        printf("\nValue inserted ");

}
void delete()
{
        int item;
        if (front == -1 || front > rear)
        {
            printf("\nUNDERFLOW\n");
            return;

        }
        else
        {
            item = queue[front];
            if(front == rear)
            {
                front = -1;
                rear = -1 ;
```

```
            }
            else
            {
                front = front + 1;
            }
            printf("\nvalue deleted ");
        }




}


void display()
{
        int i;
        if(rear == -1)
        {
            printf("\nEmpty queue\n");
        }
        else
        {       printf("\nprinting values .....\n");
            for(i=front;i<=rear;i++)
            {
                printf("\n%d\n",queue[i]);
            }
        }
}
```

OUTPUT

```
************************Main Menu****************************

==================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?1

Enter the element

2    3    4

Value inserted

************************Main Menu****************************

==================================================================

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?
```

printing values .....

2

**********Main Menu**********

==========

1.insert an element

2.Delete an element

3.Display the queue

4.Exit

Enter your choice ?

=== Code Execution Successful ===

IMPLEMENTATION OF QUEUE USING LINKED LIST

```
#include <stdio.h>

#include <stdlib.h>


struct Node {

    int data;

    struct Node* next;

};


struct Queue {
```

```
    struct Node *front, *rear;
};

struct Node* newNode(int data) {
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = data;
    temp->next = NULL;
    return temp;
}

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

void enqueue(struct Queue* queue, int data) {
    struct Node* temp = newNode(data);

    if (queue->rear == NULL) {
        queue->front = queue->rear = temp;
        return;
    }

    queue->rear->next = temp;
    queue->rear = temp;
```

```
}

void dequeue(struct Queue* queue) {
	if (queue->front == NULL)
		return;

	struct Node* temp = queue->front;
	queue->front = queue->front->next;

	if (queue->front == NULL)
		queue->rear = NULL;

	free(temp);
}

int main() {
	struct Queue* queue = createQueue();

	enqueue(queue, 10);
	enqueue(queue, 20);
	dequeue(queue);
	enqueue(queue, 30);
	dequeue(queue);
	enqueue(queue, 40);
	enqueue(queue, 50);
	dequeue(queue);
```

```
    printf("Queue Front: %d\n", queue->front->data);
    printf("Queue Rear: %d\n", queue->rear->data);

    return 0;
}
```

```
Output
/tmp/gb7BxUQ9yY.o
Queue Front: 40
Queue Rear: 50

=== Code Execution Successful ===
```

IMPLEMENTATION OF POSTFIX AND INFIX IN STACK

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

#define MAX_SIZE 100

typedef struct {
    char items[MAX_SIZE];
    int top;
} Stack;

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

```
char pop(Stack *stack) {
	return stack->items[stack->top--];
}

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

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

void infixToPostfix(char *infix, char *postfix) {
	Stack stack;
	stack.top = -1;
	int i = 0, j = 0;

	while (infix[i] != '\0') {
		if (infix[i] == '(') {
			push(&stack, infix[i]);
			i++;
		} else if (infix[i] == ')') {
```

```
                while (stack.items[stack.top] != '(') {
                        postfix[j++] = pop(&stack);
                }
                stack.top--;
                i++;
        } else if (isOperator(infix[i])) {
                while (stack.top != -1 && precedence(stack.items[stack.top]) >= precedence(infix[i])) {
                        postfix[j++] = pop(&stack);
                }
                push(&stack, infix[i]);
                i++;
        } else {
                postfix[j++] = infix[i++];
        }
    }

    while (stack.top != -1) {
        postfix[j++] = pop(&stack);
    }

    postfix[j] = '\0';
}

int main() {
    char infix[MAX_SIZE];
    char postfix[MAX_SIZE];

    printf("Enter an infix expression: ");
```

```
    scanf("%s", infix);

    infixToPostfix(infix, postfix);

    printf("Postfix expression: %s\n", postfix);

    return 0;
}
```