Assignment No : 2

2. WAP to perform stack and queue

STACK

```c
// C program for array implementation of stack
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

// A structure to represent a stack
struct Stack {
    int top;
    unsigned capacity;
    int* array;
};

// function to create a stack of given capacity. It initializes size of
// stack as 0
struct Stack* createStack(unsigned capacity)
{
    struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack));
    stack->capacity = capacity;
    stack->top = -1;
    stack->array = (int*)malloc(stack->capacity * sizeof(int));
    return stack;
}

// Stack is full when top is equal to the last index
int isFull(struct Stack* stack)
{
    return stack->top == stack->capacity - 1;
}

// Stack is empty when top is equal to -1
int isEmpty(struct Stack* stack)
{
    return stack->top == -1;
}

// Function to add an item to stack. It increases top by 1
void push(struct Stack* stack, int item)
{
    if (isFull(stack))
        return;
    stack->array[++stack->top] = item;
```

```c
    printf("%d pushed to stack\n", item);
}

// Function to remove an item from stack. It decreases top by 1
int pop(struct Stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top--];
}

// Function to return the top from stack without removing it
int peek(struct Stack* stack)
{
    if (isEmpty(stack))
        return INT_MIN;
    return stack->array[stack->top];
}

// Driver program to test above functions
int main()
{
    struct Stack* stack = createStack(100);

    push(stack, 10);
    push(stack, 20);
    push(stack, 30);

    printf("%d popped from stack\n", pop(stack));

    return 0;
}
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\vaishnavi> cd "c:\Users\vaishnavi\Desktop\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
10 pushed to stack
20 pushed to stack
30 pushed to stack
30 popped from stack
PS C:\Users\vaishnavi\Desktop>
```

```c
// C program for array implementation of queue
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

// A structure to represent a queue
struct Queue {
    int front, rear, size;
    unsigned capacity;
    int* array;
};

// function to create a queue
// of given capacity.
// It initializes size of queue as 0
struct Queue* createQueue(unsigned capacity)
{
    struct Queue* queue = (struct Queue*)malloc(
        sizeof(struct Queue));
    queue->capacity = capacity;
    queue->front = queue->size = 0;

    // This is important, see the enqueue
    queue->rear = capacity - 1;
    queue->array = (int*)malloc(
        queue->capacity * sizeof(int));
    return queue;
}

// Queue is full when size becomes
// equal to the capacity
int isFull(struct Queue* queue)
{
    return (queue->size == queue->capacity);
}

// Queue is empty when size is 0
int isEmpty(struct Queue* queue)
{
    return (queue->size == 0);
}

// Function to add an item to the queue.
// It changes rear and size
void enqueue(struct Queue* queue, int item)
{
    if (isFull(queue))
```

```
        return;
    queue->rear = (queue->rear + 1)
                % queue->capacity;
    queue->array[queue->rear] = item;
    queue->size = queue->size + 1;
    printf("%d enqueued to queue\n", item);
}

// Function to remove an item from queue.
// It changes front and size
int dequeue(struct Queue* queue)
{
    if (isEmpty(queue))
        return INT_MIN;
    int item = queue->array[queue->front];
    queue->front = (queue->front + 1)
                % queue->capacity;
    queue->size = queue->size - 1;
    return item;
}

// Function to get front of queue
int front(struct Queue* queue)
{
    if (isEmpty(queue))
        return INT_MIN;
    return queue->array[queue->front];
}

// Function to get rear of queue
int rear(struct Queue* queue)
{
    if (isEmpty(queue))
        return INT_MIN;
    return queue->array[queue->rear];
}

// Driver program to test above functions./
int main()
{
    struct Queue* queue = createQueue(1000);

    enqueue(queue, 10);
    enqueue(queue, 20);
    enqueue(queue, 30);
    enqueue(queue, 40);

    printf("%d dequeued from queue\n\n",
```

```
        dequeue(queue));

    printf("Front item is %d\n", front(queue));
    printf("Rear item is %d\n", rear(queue));

    return 0;
}
```

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\vaishnavi> cd "c:\Users\vaishnavi\Desktop\" ; if ($?) { gcc p1.c -o p1 } ; if ($?) { .\p1 }
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
10 dequeued from queue

Front item is 20
Rear item is 40
PS C:\Users\vaishnavi\Desktop>
```