Stack & Queue Array Implementation Basic Operations

<table>
<tr><th>Operations</th><th>Stack</th><th colspan="3">Queue</th></tr>
<tr><th></th><th></th><th>Queue</th><th>Circular Queue</th><th>Double Ended Queue (DeQue)</th></tr>
<tr><td>push/enqueue</td><td>

```
void push(int data)
{
  if (isFull())
  {
    printf("Stack Overflow\n");
  }
  else
  {
    top = top + 1;
    stack_arr[top] = data;
    printf("Pushed %d into the stack.\n",
data);
  }
}
```

</td><td>

```
void enqueue(int data)
{
  if (isFull())
  {
    printf("Queue Overflow\n");
    exit(1);
  }
  if (front == -1)
  {
    front = 0;
  }
  rear = rear + 1;
  queue[rear] = data;
}
```

</td><td>

```
void enqueue(int data)
{
  if (isFull())
  {
    printf("Queue Overflow\n");
    exit(1);
  }
  if (front == -1)
  {
    front = 0;
  }
  if (rear == MAX - 1)
  {
    rear = 0;
  }
  else
  {
    rear = rear + 1;
  }
  circular_queue[rear] = data;
}
```

</td><td>

```
void enqueueFront(int data)
{
  if (isFull())
  {
    printf("Queue Overflow\n");
    exit(1);
  }
  if (front == -1)
  {
    front = 0;
    rear = 0;
  }
  else if (front == 0)
  {
    front = MAX - 1;
  }
  else
  {
    front = front - 1;
  }
  deque[front] = data;
}
```

```
void enqueueRear(int data)
{
  if (isFull())
  {
    printf("Queue Overflow\n");
    exit(1);
  }
  if (front == -1)
  {
    front = 0;
    rear = 0;
  }
  else if (rear == MAX - 1)
  {
    rear = 0;
  }
  else
  {
    rear = rear + 1;
```

</td></tr>
</table>

				} deque[rear] = *data*; }

pop/dequeue	*int* pop()
	{
	if (isEmpty())
	{
	printf("Stack Underflow\n");
	exit(1);
	}
	else
	{
	int value = stack_arr[top];
	top = top - 1;
	return value;
	}
	}
		int dequeue()
		{
		int data;
		if (isEmpty())
		{
		printf("Queue Underflow\n");
		exit(1);
		}
		data = queue[front];
		{
		front++;
		}
		return data;
		}
			int dequeue()
			{
			int data;
			if (isEmpty())
			{
			printf("Queue Underflow\n");
			exit(1);
			}
			data = circular_queue[front];
			if (front == rear)
			{
			front = -1;
			rear = -1;
			}
			else if (front == MAX - 1)
			{
			front = 0;
			}
			else
			front = front + 1;
			return data;
			}
				int dequeueFront()
				{
				int data;
				if (isEmpty())
				{
				printf("Queue Underflow\n");
				exit(1);
				}
				data = deque[front];
				if (front == rear)
				{
				front = -1;
				rear = -1;
				}
				else if (front == MAX - 1)
				{
				front = 0;
				}
				else
				front = front + 1;
				return data;
				}

				int dequeueRear()
				{
				int data;
				if (isEmpty())
				{
				printf("Queue Underflow\n");
				exit(1);
				}
				data = deque[rear];
				if (front == rear)
				{
				front = -1;
				rear = -1;
				}
				else if (rear == 0)
				{
				rear = MAX - 1;
				}
				else
				rear = rear - 1;
				return data;
				}

isFull	`int isFull()`
	`{`
	`  if (top == MAX - 1)`
	`  {`
	`  return 1;`
	`  }`
	`  else`
	`  {`
	`  return 0;`
	`  }`
	`}`
		`int isFull()`
		`{`
		`  if (rear == MAX - 1)`
		`  {`
		`  return 1;`
		`  }`
		`  else`
		`  {`
		`  return 0;`
		`  }`
		`}`
			`int isFull()`
			`{`
			`  if ((front == 0 && rear == MAX - 1) || (front == rear + 1))`
			`  {`
			`  return 1;`
			`  }`
			`  else`
			`  {`
			`  return 0;`
			`  }`
			`}`	

isEmpty	`int isEmpty()`
	`{`
	`  if (top == -1)`
	`  {`
	`  return 1;`
	`  }`
	`  else`
	`  {`
	`  return 0;`
	`  }`
	`}`
		`int isEmpty()`
		`{`
		`  if (front == -1 || front == rear + 1)`
		`  {`
		`  return 1;`
		`  }`
		`  else`
		`  {`
		`  return 0;`
		`  }`
		`}`
			`int isEmpty()`
			`{`
			`  if (front == -1)`
			`  {`
			`  return 1;`
			`  }`
			`  else`
			`  {`
			`  return 0;`
			`  }`
			`}`	

peek	`int peek()`
	`{`
	`  if (isEmpty())`
	`  {`
	`  printf("Stack Underflow\n");`
	`  exit(1);`
	`  }`
	`  else`
	`  {`
	`  return stack_arr[top];`
	`  }`
	`}`
		`int peek()`
		`{`
		`  if (isEmpty())`
		`  {`
		`  printf("Queue Underflow\n");`
		`  exit(1);`
		`  }`
		`  return queue[front];`
		`}`
			`int peek()`
			`{`
			`  if (isEmpty())`
			`  {`
			`  printf("Queue Underflow\n");`
			`  exit(1);`
			`  }`
			`  return circular_queue[front];`
			`}`
				`int peek()`
				`{`
				`  if (isEmpty())`
				`  {`
				`  printf("Queue Underflow\n");`
				`  exit(1);`
				`  }`
				`  return deque[front];`
				`}`

print				

```c
void print()
{
  if (isEmpty())
  {
    printf("Stack is empty\n");
    return;
  }
  else
  {
    printf("Stack elements :\n\n");
    for (int i = top; i >= 0; i--)
      printf("%d\n", stack_arr[i]);
    printf("\n");
  }
}
```

```c
void print()
{
  int i;
  if (isEmpty())
  {
    printf("Queue Underflow\n");
    exit(1);
  }
  printf("Queue: ");
  for (i = front; i <= rear; i++)
  {
    printf("%d ", queue[i]);
  }
  printf("\n");
}
```

```c
void print()
{
  int temp;
  if (isEmpty())
  {
    printf("Queue Underflow\n");
    exit(1);
  }
  temp = front;
  if (front <= rear)
  {
    while (temp <= rear)
    {
      printf("%d ", circular_queue[temp]);
      temp++;
    }
  }
  else
  {
    while (temp <= MAX - 1)
    {
      printf("%d ", circular_queue[temp]);
      temp++;
    }
    temp = 0;
    while (temp <= rear)
    {
      printf("%d ", circular_queue[temp]);
      temp++;
    }
  }
  printf("\n");
}
```

```c
void print()
{
  int temp;
  if (isEmpty())
  {
    printf("Queue Underflow\n");
    exit(1);
  }
  temp = front;
  if (front <= rear)
  {
    while (temp <= rear)
    {
      printf("%d ", deque[temp]);
      temp++;
    }
  }
  else
  {
    while (temp <= MAX - 1)
    {
      printf("%d ", deque[temp]);
      temp++;
    }
    temp = 0;
    while (temp <= rear)
    {
      printf("%d ", deque[temp]);
      temp++;
    }
  }
  printf("\n");
}
```