

```
1 #include <stdio.h>
2 #define SIZE 5
3 void enQueue(int);
4 void deQueue();
5 void display();
6 int items[SIZE], front = -1, rear = -1;
7 int main()
8 {
9     deQueue();
10    enQueue(1);
11    enQueue(2);
12    enQueue(3);
13    enQueue(4);
14    enQueue(5);
15    enQueue(6);
16    display();
17    deQueue();
18    display();
19    return 0;
20 }
21 void enQueue(int value)
22 {
23     if (rear == SIZE - 1)
24         printf("\nQueue is Full!!");
25     else
26     {
```

/tmp/1myZhC1vNI.o

Queue is Empty!!

Inserted -> 1

Inserted -> 2

Inserted -> 3

Inserted -> 4

Inserted -> 5

Queue is Full!!

Queue elements are:

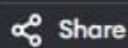
1 2 3 4 5

Deleted : 1

Queue elements are:

2 3 4 5

=== Code Execution Successful ===



```
1  #include <stdio.h>
2  void insertElement(int arr[], int n, int pos, int value) {
3      for (int i = n; i > pos; i--) {
4          arr[i] = arr[i - 1];
5      }
6      arr[pos] = value;
7  }
8  void deleteElement(int arr[], int n, int pos) {
9      for (int i = pos; i < n - 1; i++) {
10         arr[i] = arr[i + 1];
11     }
12 }
13 int main() {
14     int arr[100] = {1, 2, 3, 4, 5};
15     int n = 5;
16     insertElement(arr, n, 2, 99);
17     n++;
18     printf("Array after insertion: ");
19     for (int i = 0; i < n; i++) {
20         printf("%d ", arr[i]);
21     }
22     printf("\n");
23     deleteElement(arr, n, 3);
24     n--;
25     printf("Array after deletion: ");
26     for (int i = 0; i < n; i++) {
```

/tmp/3eNlbcXrdq.o

Array after insertion: 1 2 99 3 4 5

Array after deletion: 1 2 99 4 5

=== Code Execution Successful ===



```
1 #include <limits.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 struct Stack
5 {
6     int top;
7     unsigned capacity;
8     int* array;
9 };
10 struct Stack* createStack(unsigned capacity)
11 {
12     struct Stack* stack = (struct Stack*)malloc(sizeof(struct Stack
13     ));
14     stack->capacity = capacity;
15     stack->top = -1;
16     stack->array = (int*)malloc(stack->capacity * sizeof(int));
17     return stack;
18 }
19 int isFull(struct Stack* stack)
20 {
21     return stack->top == stack->capacity - 1;
22 }
23 int isEmpty(struct Stack* stack)
24 {
25     return stack->top == -1;
```

```
/tmp/11YJBRj0eY.o
10 pushed to stack
20 pushed to stack
30 pushed to stack
30 popped from stack
```

=== Code Execution Successful ===