DSA – Practical: Stacks and Queues

1. Implement the basic operations of a stack of elements using a dynamic array structure.

Use the structures below and implement the given functions:

```
typedef struct Element {
       int key;
       /*possible other fields*/

} Element;
```

```
typedef struct Stack{
       Element* stackData;
       int top;
       int capacity;
} Stack;
```

```
Stack* createEmptyStack () //creates empty stack, initial capacity of 10 elements
void pushElement (Stack* s, Element e) //pushes e to the top of the stack
Element popElement (Stack* s) //pops the element on the top and returns it
bool isEmpty (Stack* s) //tells whether the stack s is empty or not
int elementsCount (Stack* s) //returns how many elements are inside the stack
```

Test the operations by creating a stack in the main() function and apply different operations to it:

```
void main () {
   Stack* myStack = createEmptyStack ();
   Element e;
   e.key = 99;   pushElement (myStack, e);
   e.key = 88;   pushElement (myStack, e);
   e.key = 110;  pushElement (myStack, e);
   ...

   while(! isEmpty(myStack))
      printf("Key of element at stack top is %d \n", popElement (myStack) .key );

   free(myStack -> stackData);
   free(myStack);
   system("pause");
}
```

Hint: If *bool* type is not defined in your C compiler (like in VisualStudio), just include this line in your code: `typedef enum { false, true } bool;`



2. Implement the basic operations of a Queue of elements using linked list structure.

Use the structures below and implement the given functions:

```
typedef struct QueueNode {
       Element  elem;
       struct QueueNode * next;

} QueueNode;
```

```
typedef struct Queue{
       QueueNode * first;
       QueueNode * last;
       int size;
} Queue;
```

```
Queue* createEmptyQueue () //creates empty queue
void addToQueue (Queue* q, Element e) //adds e to the end of the queue
Element removeFromQueue (Queue* q) //removes element at beginning and returns it
bool isQueueEmpty (Queue* q) //tells whether the queue q is empty or not
int queueSize (Queue* q) //returns how many elements are inside the queue
```

Test the operations by creating a queue in the main() function and apply different operations to it:

```
void main () {
   Queue* myQueue = createEmptyQueue ();
   Element e;
   e.key = 99;   addToQueue (myQueue, e);
   e.key = 88;   addToQueue (myQueue, e);
   e.key = 110;  addToQueue (myQueue, e);
   ...

   while(! isQueueEmpty(myQueue))
      printf("Key of element at queue front is %d\n", removeFromQueue(myQueue) .key);

   free(myQueue);
   system("pause");
}
```

- The End -