Stack Implementation in C

Problem Statement: Implement a stack using an array in C and perform basic stack operations such as push, pop, peek, check if full, and check if empty.

Objective: - Learn the concept of stack data structure. - Implement stack operations using arrays. - Understand stack behavior (LIFO: Last In First Out).

Algorithm:

1. **Push Operation:**
    - Check if the stack is full using `isfull()`.
    - If not full, increment `top` and insert the element at `stack[top]`.
    - If full, display “Stack is full”.
2. **Pop Operation:**
    - Check if the stack is empty using `isempty()`.
    - If not empty, retrieve the element at `stack[top]`.
    - Decrement `top` to remove the element.
    - If empty, display “Stack is empty”.
3. **Peek Operation:**
    - Return the element at `stack[top]` without removing it.
4. **isEmpty Operation:**
    - If `top == -1`, the stack is empty.
5. **isFull Operation:**
    - If `top == MAXSIZE - 1`, the stack is full.

Program Code:

```c
#include <stdio.h>
int MAXSIZE = 8;
int stack[8];
int top = -1;

int isempty(){
   if(top == -1)
      return 1;
   else
      return 0;
}
int isfull(){
   if(top == MAXSIZE - 1)
      return 1;
   else
      return 0;
}
int peek(){
   return stack[top];
```

```
}
int pop(){
   int data;
   if(!isempty()) {
      data = stack[top];
      top = top - 1;
      return data;
   } else {
      printf("Could not retrieve data, Stack is empty.\n");
   }
}
int push(int data){
   if(!isfull()) {
      top = top + 1;
      stack[top] = data;
   } else {
      printf("Could not insert data, Stack is full.\n");
   }
}
int main(){
   push(44);
   push(10);
   push(62);
   push(123);
   push(15);

   printf("Element at top of the stack: %d\n", peek());
   printf("Elements: \n");

   while(!isempty()) {
      int data = pop();
      printf("%d\n", data);
   }

   printf("Stack full: %s\n", isfull()?"true":"false");
   printf("Stack empty: %s\n", isempty()?"true":"false");
   return 0;
}
```

Sample Output:

```
Element at top of the stack: 15
Elements:
15
123
62
10
44
Stack full: false
Stack empty: true
```