C program to reverse a stack data structure using recursion

```
#include <stdio.h>

#define MAXSIZE 7
#define TRUE 1
#define FALSE 0

// Structure defining Stack data structure
struct Stack {
    int top;
    int array[MAXSIZE];
} st;

/*Initializes the top index to -1 */

void initialize() {
 st.top = -1;
}

/*Checks if Stack is Full or not */
int isFull() {
    if(st.top >= MAXSIZE-1)
        return TRUE;
    else
        return FALSE;
 }

 /*Checks if Stack is Empty or not*/

 int isEmpty() {
  if(st.top == -1)
      return TRUE;
  else
      return FALSE;
 }

 /* Adds an element to stack and then increment top index */

 void push(int num) {
     if (isFull())
         printf("Stack is Full...\n");
     else {
         st.array[st.top + 1] = num;
         st.top++;
     }
 }

 /*
  Removes top element from stack and decrement top index */

 int pop() {
     if (isEmpty())
         printf("Stack is Empty...\n");
```

```
    else {
     st.top = st.top - 1;
        return st.array[st.top+1];
    }
}

/*
 Prints elements of stack using recursion */

void printStack(){
 if(!isEmpty()){
     int temp = pop();
     printStack();
     printf(" %d ", temp);
     push( temp);
    }
}
void insertAtBottom(int item) {
    if (isEmpty()) {
       push(item);
    } else {

        /* Store the top most element of stack in top variable and
        recursively call insertAtBottom for rest of the stack */
        int top = pop();
        insertAtBottom(item);

        /* Once the item is inserted at the bottom, push the
        top element back to stack */
        push(top);
    }
}
                          → reverses a stack
void reverse() {
    if (!isEmpty()) {
        /* keep on popping top element of stack in
        every recursive call till stack is empty  */
        int top = pop();
        reverse();

        /* Now, instead of inserting element back on top
        of stack, we will insert it at the bottom of stack */
        insertAtBottom(top);
    }
}
/*
Returns the number of elements in Stack
*/
int getSize(){
 return st.top+1;
}

int main() {
 /* Initializing top index of stack */
   initialize(st);
    /* Adding elements in stack */
    push(1);
    push(2);
```

```
    push(3);
    push(4);
    push(5);
    printf("Original Stack\n");
    printStack();
    reverse();
    printf("\nReversed Stack\n");
    printStack();
    return 0;
}
```

Output

```
Original Stack
 1 2 3 4 5
Reversed Stack
 5 4 3 2 1
```