Manav Rachna International Institute of Research and Studies

Bachelor's in computer applications

Data Structures using C

Submitted by: Urvashi Pahuja

Department: School of Computer Applications

Course: Bachelor's in computer applications

Roll No: 24/SCA/BCA/052

Semester: 2nd

Subject: Data Structures using C

DS File

1. Push operation in stacks

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 5


struct Stack {
   int arr[MAX];
   int top;
};


void initStack(struct Stack* stack) {
   stack->top = -1;
}


int isFull(struct Stack* stack) {
   return stack->top == MAX - 1;
}


void push(struct Stack* stack, int value) {
   if (isFull(stack)) {
      printf("Stack Overflow! Cannot push %d\n", value);
   } else {
      stack->arr[++(stack->top)] = value;
      printf("%d pushed to stack\n", value);
   }
}


void printStack(struct Stack* stack) {
   if (stack->top == -1) {
      printf("Stack is empty\n");
   } else {
      printf("Stack elements: ");
```

```
        for (int i = 0; i <= stack->top; i++) {
            printf("%d ", stack->arr[i]);
        }
        printf("\n");
    }
}

int main() {
    struct Stack stack;
    initStack(&stack);


    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);
    push(&stack, 40);
    push(&stack, 50);
    push(&stack, 60);


    printStack(&stack);

    return 0;
}
```

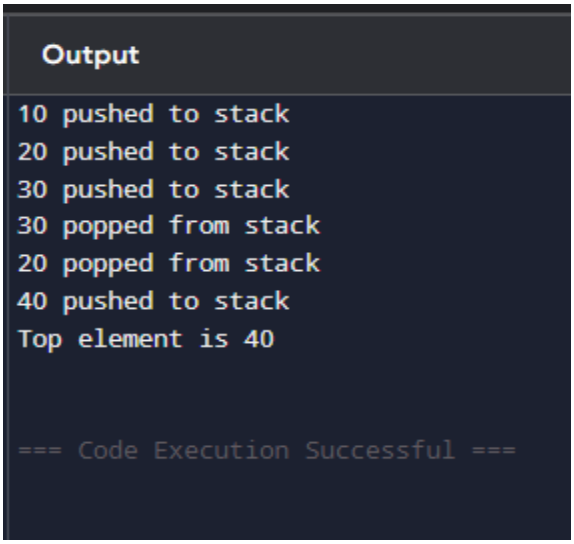
Output

```
10 pushed to stack
20 pushed to stack
30 pushed to stack
40 pushed to stack
50 pushed to stack
Stack Overflow! Cannot push 60
Stack elements: 10 20 30 40 50


=== Code Execution Successful ===
```

2. Pop operation in stacks

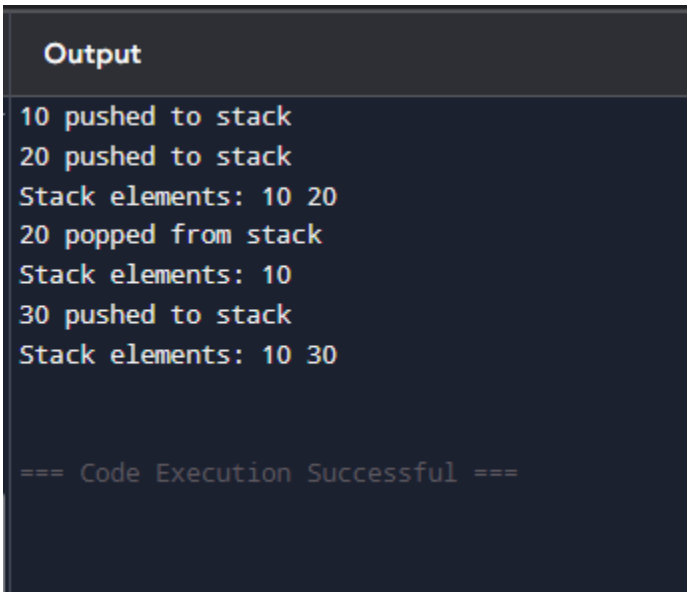
```
 #include <stdio.h>
#include <stdlib.h>

#define MAX 5


struct Stack {
   int arr[MAX];
   int top;
};


void initialize(struct Stack *s) {
   s->top = -1;
}


int isFull(struct Stack *s) {
   return s->top == MAX - 1;
}


int isEmpty(struct Stack *s) {
   return s->top == -1;
}


void push(struct Stack *s, int value) {
   if (isFull(s)) {
      printf("Stack Overflow\n");
   } else {
      s->arr[++(s->top)] = value;
      printf("%d pushed to stack\n", value);
   }
}

int pop(struct Stack *s) {
   if (isEmpty(s)) {
      printf("Stack Underflow\n");
      return -1;
   } else {
      int poppedValue = s->arr[s->top--];
      return poppedValue;
   }
}


int peek(struct Stack *s) {
   if (isEmpty(s)) {
      printf("Stack is empty\n");
```

```
        return -1;
    }
    return s->arr[s->top];
}

int main() {
    struct Stack stack;
    initialize(&stack);

    push(&stack, 10);
    push(&stack, 20);
    push(&stack, 30);

    printf("%d popped from stack\n", pop(&stack));
    printf("%d popped from stack\n", pop(&stack));

    push(&stack, 40);
    printf("Top element is %d\n", peek(&stack));

}
```



3. IMPLEMENTATION OF STACK BY USING ARRAY

```
#include <stdio.h>
#define MAX 5

int stack[MAX], top = -1;

void push(int x) {
   if (top == MAX - 1) {
      printf("Stack Overflow!\n");
   } else {
      top++;
      stack[top] = x;
      printf("%d pushed to stack\n", x);
   }
}

void pop() {
   if (top == -1) {
      printf("Stack Underflow!\n");
   } else {
      printf("%d popped from stack\n", stack[top]);
      top--;
   }
}

void display() {
   if (top == -1) {
      printf("Stack is empty\n");
   } else {
      printf("Stack elements: ");
      for (int i = 0; i <= top; i++) {
         printf("%d ", stack[i]);
      }
      printf("\n");
   }
}

int main() {
   push(10);
   push(20);
   display();
   pop();
   display();
   push(30);
   display();
   return 0;
}
```


Output
10 pushed to stack
20 pushed to stack
Stack elements: 10 20
20 popped from stack
Stack elements: 10
30 pushed to stack
Stack elements: 10 30
=== Code Execution Successful ===