**Experiment No 1**

**Name -: Aditya Babaso Birangaddi**

**Code -:**

#include <limits.h>

#include <stdio.h>

#include <stdlib.h>

// A structure to represent a stack

struct Stack {

int top;

unsigned capacity;

int\* array;

};

// function to create a stack of given capacity. It initializes size of

// stack as 0

struct Stack\* createStack(unsigned capacity)

{

struct Stack\* stack = (struct Stack\*)malloc(sizeof(struct Stack));

stack->capacity = capacity;

stack->top = -1;

stack->array = (int\*)malloc(stack->capacity \* sizeof(int));

return stack;

}

// Stack is full when top is equal to the last index

int isFull(struct Stack\* stack)

{

return stack->top == stack->capacity - 1;

}

// Stack is empty when top is equal to -1

int isEmpty(struct Stack\* stack)

{

return stack->top == -1;

}

// Function to add an item to stack. It increases top by 1

void push(struct Stack\* stack, int item)

{

if (isFull(stack))

return;

stack->array[++stack->top] = item;

printf("%d pushed to stack\n", item);

}

// Function to remove an item from stack. It decreases top by 1

int pop(struct Stack\* stack)

{

if (isEmpty(stack))

return INT\_MIN;

return stack->array[stack->top--];

}

// Function to return the top from stack without removing it

int peek(struct Stack\* stack)

{

if (isEmpty(stack))

return INT\_MIN;

return stack->array[stack->top];

}

// Driver program to test above functions

int main()

{

struct Stack\* stack = createStack(100);

push(stack, 10);

push(stack, 20);

push(stack, 30);

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

return 0;

}

**Output -:**

