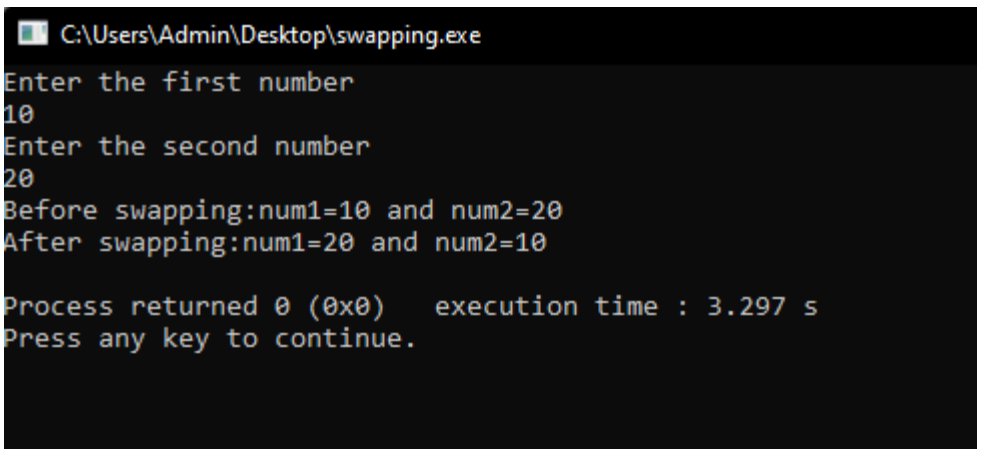
LAB PROGRAM 1: SWAPPING 2 NUMBERS USING POINTERS WITH A FUNCTION

SOURCE CODE:

```
#include<stdio.h>
void swap(int *a,int *b);
int main()
{
int num1,num2;
printf("Enter the first number\n");
scanf("%d",&num1);
printf("Enter the second number\n");
scanf("%d",&num2);
printf("Before swapping:num1=%d and num2=%d\n",num1,num2);
swap(&num1,&num2);
printf("After swapping:num1=%d and num2=%d\n",num1,num2);
return(0);
}
void swap(int *a,int *b)
{
int temp=*a;
*a=*b;
*b=temp;
}
```

OUTPUT:



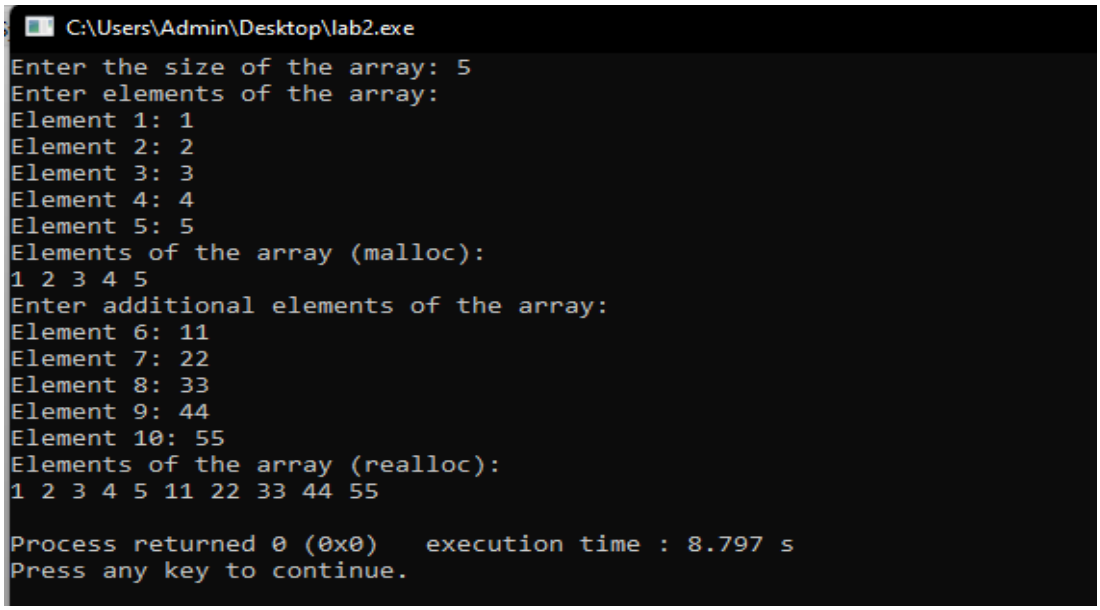
LAB PROGRAM 2: WRITE A PROGRAM TO IMPLEMENT DYNAMIC MEMORY ALLOCATION LIKE MALLOC, CALLOC, FREE AND REALLOC

SOURCE CODE:

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

void* myMalloc(size_t size) {
   return malloc(size);
}

void* myRealloc(void* ptr, size_t size) {
   return realloc(ptr, size);
}

void* myCalloc(size_t num, size_t size) {
   return calloc(num, size);
}

void myFree(void* ptr) {
   free(ptr);
}

int main() {
   int *arr1, *arr2;
   size_t size;

   printf("Enter the size of the array: ");
   scanf("%zu", &size);


   arr1 = (int*)myMalloc(size * sizeof(int));

   if (arr1 == NULL) {
      printf("Memory allocation failed.\n");
      return 1;

   }

   printf("Enter elements of the array:\n");
   for (size_t i = 0; i < size; i++) {
      printf("Element %zu: ", i + 1);
      scanf("%d", &arr1[i]);
   }

   printf("Elements of the array (malloc):\n");
   for (size_t i = 0; i < size; i++) {
      printf("%d ", arr1[i]);
```

```
    }
    printf("\n");


    size *= 2;
    arr2 = (int*)myRealloc(arr1, size * sizeof(int));


    if (arr2 == NULL) {
        printf("Memory reallocation failed.\n");
        myFree(arr1);
        return 1;
    }

    printf("Enter additional elements of the array:\n");
    for (size_t i = size / 2; i < size; i++) {
        printf("Element %zu: ", i + 1);
        scanf("%d", &arr2[i]);
    }

    printf("Elements of the array (realloc):\n");
    for (size_t i = 0; i < size; i++) {
        printf("%d ", arr2[i]);
    }
    printf("\n");

    myFree(arr2);

    return 0;
}
```

OUTPUT:



LAB PROGRAM 3: Write a program to simulate the working of stack using an array with the following:

a) Push

b) Pop

c) Display

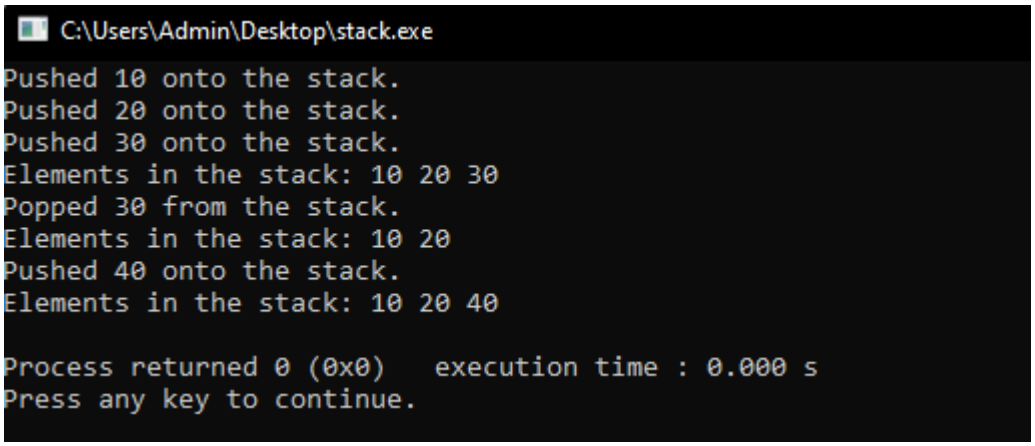
The program should print appropriate messages for stack overflow, stack underflow

SOURCE CODE:

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

#define MAX_SIZE 10

struct Stack {
   int items[MAX_SIZE];
   int top;
};


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


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


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


void push(struct Stack *stack, int value) {
   if (isFull(stack)) {
      printf("Stack overflow. Cannot push %d.\n", value);
   } else {
      stack->top++;
      stack->items[stack->top] = value;
      printf("Pushed %d onto the stack.\n", value);
   }
}

int pop(struct Stack *stack) {
```

```
    int poppedValue = -1;
    if (isEmpty(stack)) {
        printf("Stack underflow. Cannot pop from an empty stack.\n");
    } else {
        poppedValue = stack->items[stack->top];
        stack->top--;
        printf("Popped %d from the stack.\n", poppedValue);
    }

    return poppedValue;
}

void display(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack is empty.\n");
    } else {
        printf("Elements in the stack: ");
        for (int i = 0; i <= stack->top; i++) {
            printf("%d ", stack->items[i]);
        }
        printf("\n");
    }
}

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

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

    pop(&stack);
    display(&stack);

    push(&stack, 40);
    display(&stack);

    return 0;
}
```

OUTPUT:


C:\Users\Admin\Desktop\stack.exe
Pushed 10 onto the stack.
Pushed 20 onto the stack.
Pushed 30 onto the stack.
Elements in the stack: 10 20 30
Popped 30 from the stack.
Elements in the stack: 10 20
Pushed 40 onto the stack.
Elements in the stack: 10 20 40
Process returned 0 (0x0) execution time : 0.000 s
Press any key to continue.