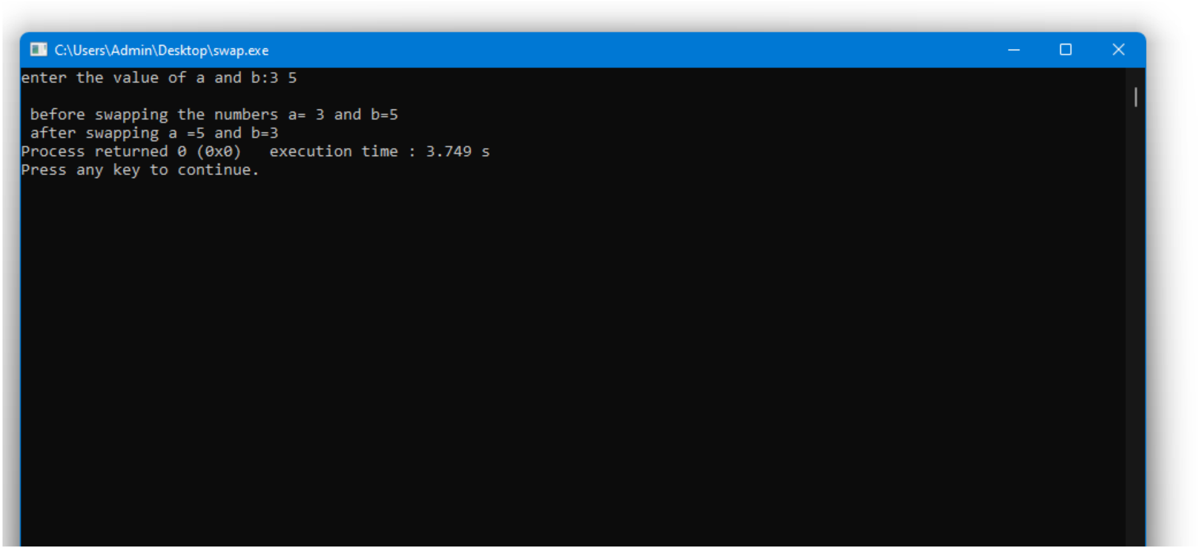
DATA STRUCTURES WITH C

Write a program with a function to swap two numbers using pointers

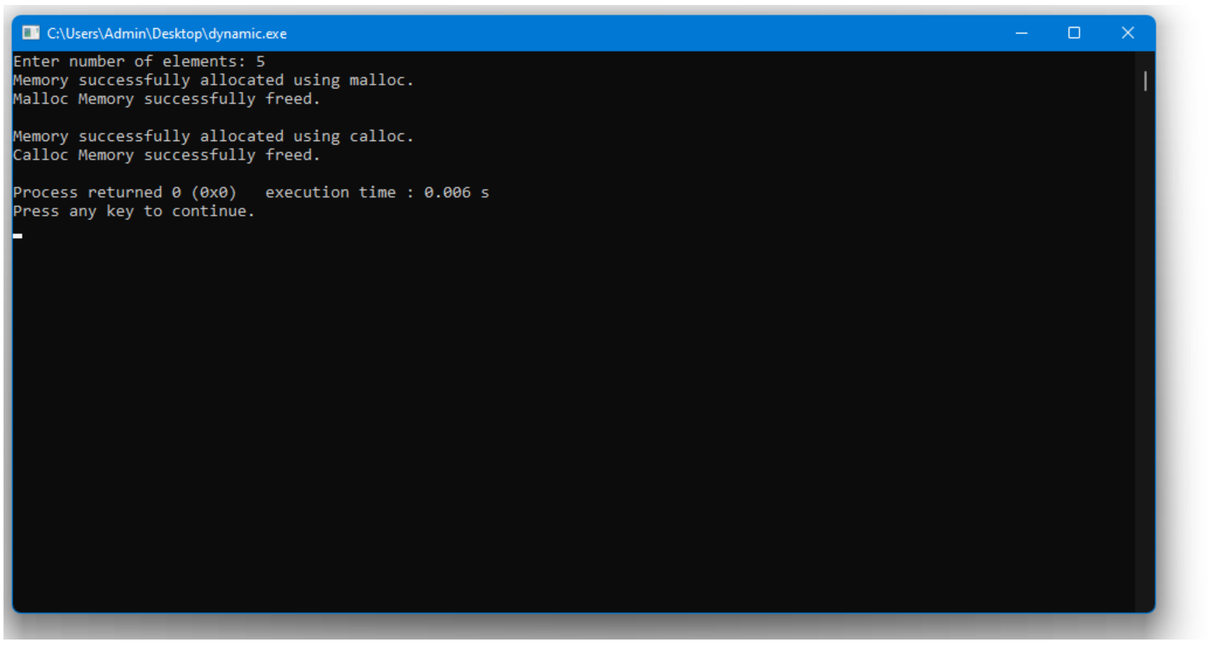
```
#include <stdio.h>

int main() {

    int a, b, temp;

    int *ptr1, *ptr2;

    printf("Enter the value of a and b: ");

    scanf("%d %d", &a, &b);

    printf("\nBefore swapping a = %d and b = %d", a, b);

    ptr1 = &a;

    ptr2 = &b;

    temp = *ptr1;

    *ptr1 = *ptr2;

    *ptr2 = temp;

    printf("\nAfter swapping a = %d and b = %d", a, b);

    return 0;

}
```



```
enter the value of a and b:3 5

 before swapping the numbers a= 3 and b=5
 after swapping a =5 and b=3
Process returned 0 (0x0)   execution time : 3.749 s
Press any key to continue.
```

Write a program to implement dynamic memory allocation functions like malloc, calloc, free, realloc

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        int *ptr, *ptr1;
        int n, i;
        n = 5;
        printf("Enter number of elements: %d\n", n);
        ptr = (int*)malloc(n * sizeof(int));
        ptr1 = (int*)calloc(n, sizeof(int));
        if (ptr == NULL || ptr1 == NULL) {
                printf("Memory not allocated.\n");
                exit(0);
        }
        else {
                printf("Memory successfully allocated using malloc.\n");
                free(ptr);
                printf("Malloc Memory successfully freed.\n");
                printf("\nMemory successfully allocated using calloc.\n");
                free(ptr1);
                printf("Calloc Memory successfully freed.\n");
        }
        return 0;
}
```

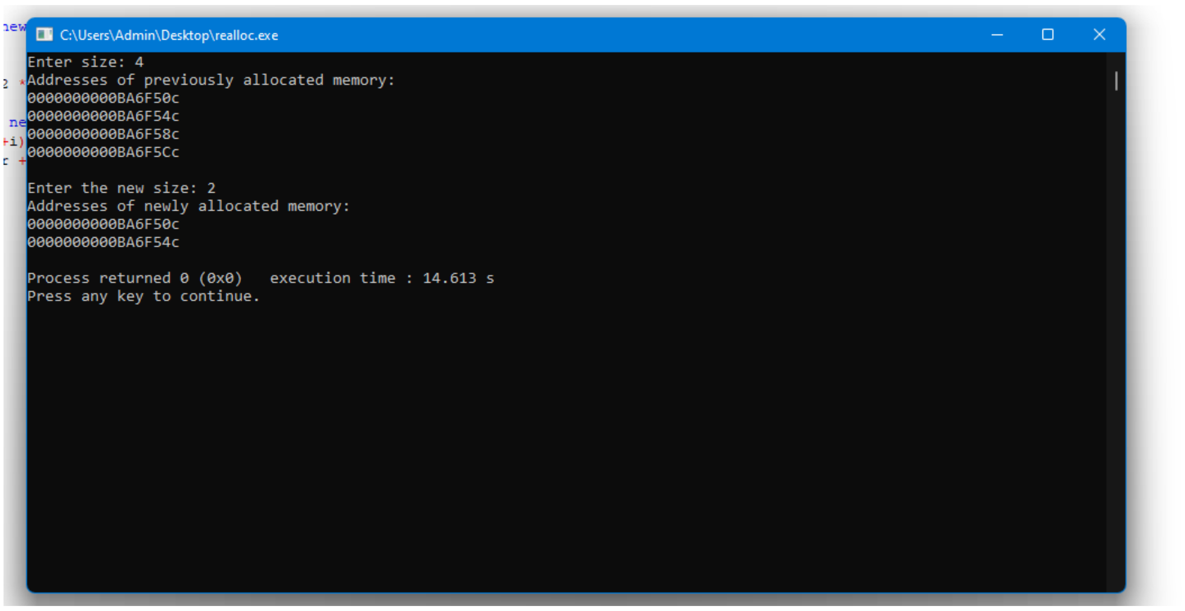


```
C:\Users\Admin\Desktop\dynamic.exe
Enter number of elements: 5
Memory successfully allocated using malloc.
Malloc Memory successfully freed.

Memory successfully allocated using calloc.
Calloc Memory successfully freed.

Process returned 0 (0x0)   execution time : 0.006 s
Press any key to continue.
```

Realloc:-

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
                    int* ptr;
                    int n, i;
                    n = 5;
                    printf("Enter number of elements: %d\n", n);
                    ptr = (int*)calloc(n, sizeof(int));
```

```
        if (ptr == NULL) {
            printf("Memory not allocated.\n");
            exit(0);
        }
        else {
            printf("Memory successfully allocated using calloc.\n");
            for (i = 0; i < n; ++i) {
                    ptr[i] = i + 1;
            }
            printf("The elements of the array are: ");
            for (i = 0; i < n; ++i) {
                    printf("%d, ", ptr[i]);
            }
            n = 10;
            printf("\n\nEnter the new size of the array: %d\n", n);
            ptr = (int*)realloc(ptr, n * sizeof(int));
            printf("Memory successfully re-allocated using realloc.\n");
            for (i = 5; i < n; ++i) {
                    ptr[i] = i + 1;
            }
            printf("The elements of the array are: ");
            for (i = 0; i < n; ++i) {
                    printf("%d, ", ptr[i]);
            }
            free(ptr);
        }
        return 0;
}
```

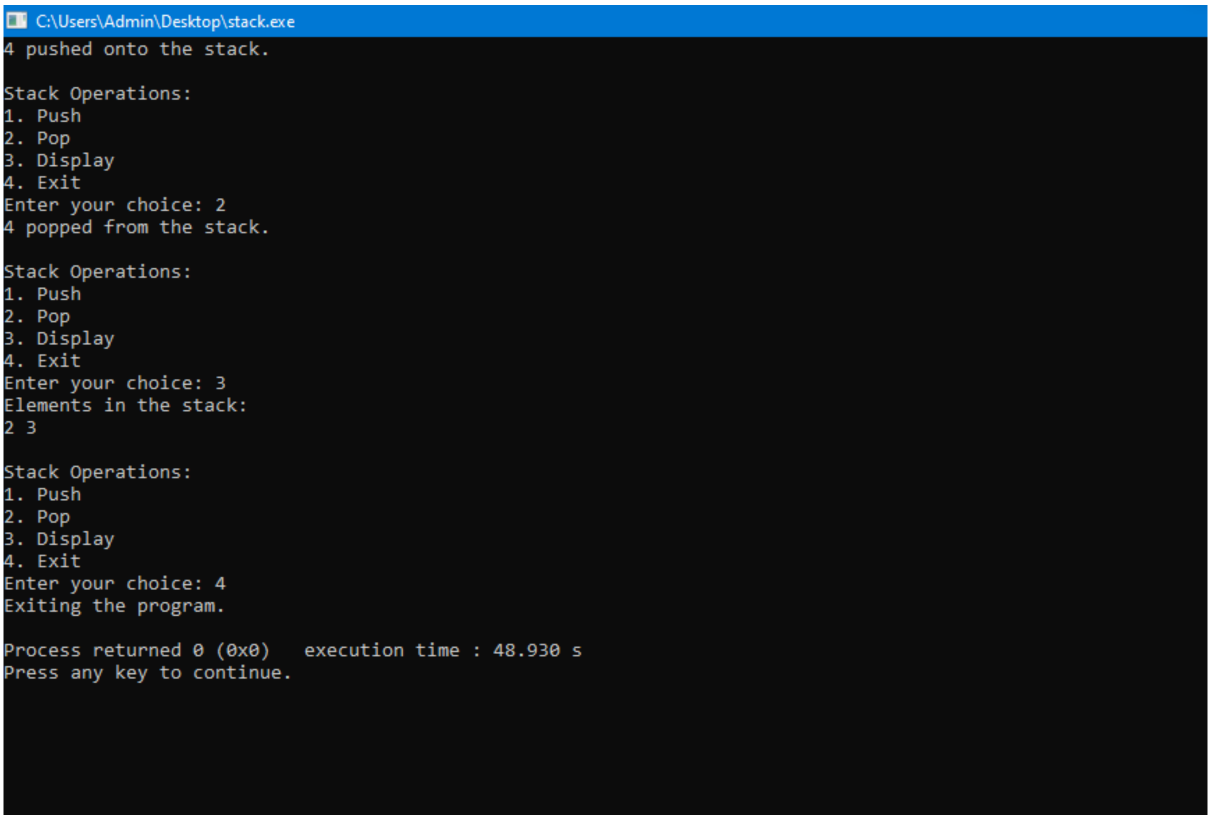


```
Enter size: 4
Addresses of previously allocated memory:
0000000000BA6F50c
0000000000BA6F54c
0000000000BA6F58c
0000000000BA6F5Cc

Enter the new size: 2
Addresses of newly allocated memory:
0000000000BA6F50c
0000000000BA6F54c

Process returned 0 (0x0)   execution time : 14.613 s
Press any key to continue.
```

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

a. Push
b. Pop
c. Display

```
#include <stdio.h>
#define SIZE 5
int stack[SIZE];
int top = -1;
void push(int element);
void pop();
void display();
int main() {
   int choice, element;
   do {

      printf("\nStack Operations:\n");
      printf("1. Push\n");
      printf("2. Pop\n");
      printf("3. Display\n");
      printf("4. Exit\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

      switch (choice) {
         case 1:
                printf("Enter the element to push: ");
                scanf("%d", &element);
                push(element);
                break;
            case 2:

                pop();
                break;
            case 3:

                display();
                break;
            case 4:

                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice. Please enter a valid option.\n");
        }
    } while (choice != 4);

    return 0;
}

void push(int element) {
    if (top == SIZE - 1) {
        printf("Stack Overflow! Cannot push element.\n");
    } else {
        top++;
        stack[top] = element;
        printf("%d pushed onto the stack.\n", element);
    }
}

void pop() {
    if (top == -1) {
        printf("Stack Underflow! Cannot pop element.\n");
    } else {
        printf("%d popped from the stack.\n", stack[top]);
        top--;
    }
}

void display() {
    if (top == -1) {
        printf("Stack is empty. Nothing to display.\n");
    } else {
        printf("Elements in the stack:\n");
        for (int i = 0; i <= top; i++) {
            printf("%d ", stack[i]);
        }
        printf("\n");
    }
}
```



C:\Users\Admin\Desktop\stack.exe

```
4 pushed onto the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 2
4 popped from the stack.

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 3
Elements in the stack:
2 3

Stack Operations:
1. Push
2. Pop
3. Display
4. Exit
Enter your choice: 4
Exiting the program.

Process returned 0 (0x0)   execution time : 48.930 s
Press any key to continue.
```