**DATA STRUCTURES-LAB 1 (21/12/23)**

1. **Develop a C program to swap two numbers using pointers.**

**CODE:**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

void swap(int \*x,int \*y)

{

    int temp;

    temp=\*x;

    \*x=\*y;

    \*y=temp;

}

void main()

{

    int a,b;

    printf("Enter two numbers: \n");

    scanf("%d%d",&a,&b);

    printf("Numbers before swapping: \n");

    printf("%d %d",a,b);

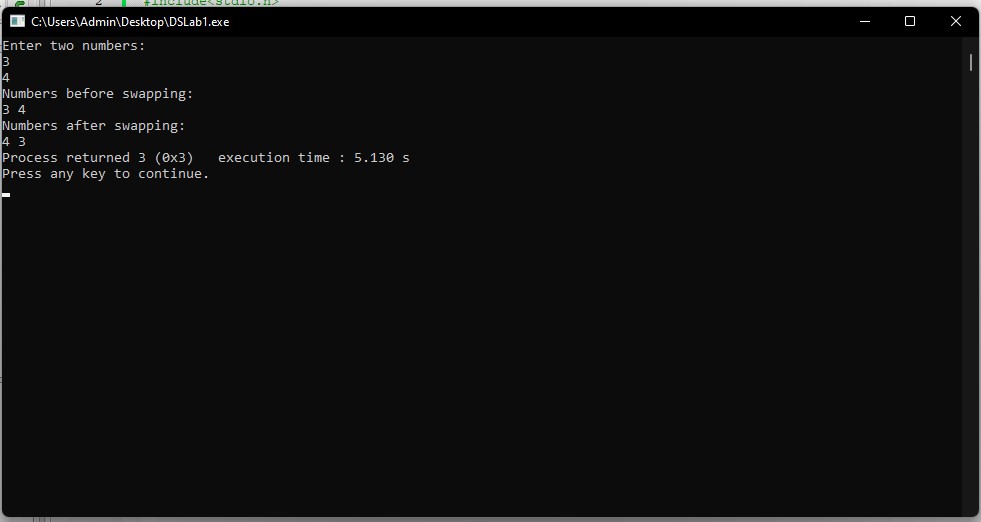
    swap(&a,&b);

    printf("\nNumbers after swapping: \n");

    printf("%d %d",a,b);

}

**OUTPUT:**

**:**

1. **Develop a C program to implement dynamic memory allocation for an array.**

**CODE:**

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

int main()

{

    // This pointer will hold the

    // base address of the block created

    int\* ptr;

    int n, i;

    n = 5;

    printf("Enter 5 elements: %d\n");

    ptr = (int\*)calloc(n, sizeof(int));

    // Check if the memory has been successfully

    // allocated by calloc or not

    if (ptr == NULL)

    {

        printf("Memory not allocated.\n");

        exit(0);

    }

    else

    {

        printf("Memory successfully allocated using calloc.\n");

        // Get the elements of the array

        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]);

        }

        // Get the new size for the array

        n = 10;

        printf("\n\nEnter the new size of the array: %d\n", n);

        // Dynamically re-allocate memory using realloc()

        ptr = (int\*)realloc(ptr, n \* sizeof(int));

        // Memory has been successfully allocated

        printf("Memory successfully re-allocated using realloc.\n");

        // Get the new elements of the array

        for (i = 5; i < n; ++i)

{

            ptr[i] = i + 1;

        }

        // Print the elements of the array

        printf("The elements of the array are: ");

        for (i = 0; i < n; ++i)

{

            printf("%d, ", ptr[i]);

        }

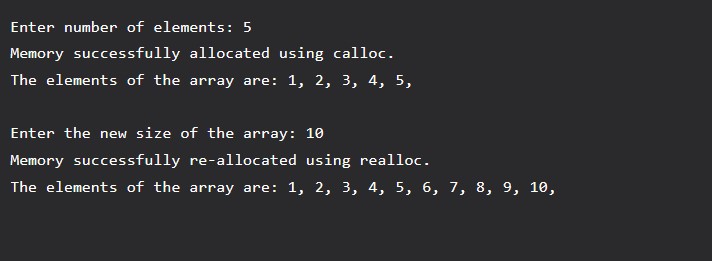
        free(ptr);

    }

    return 0;

}

**OUTPUT:**



1. **Develop a C program for stack implementation using Arrays.**

**CODE:**

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

    top=-1;

    printf("\n Enter the size of STACK[MAX=100]:");

    scanf("%d",&n);

    printf("\n\t STACK OPERATIONS USING ARRAY");

    printf("\n\t--------------------------------");

    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

    do

    {

        printf("\n Enter the Choice:");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

                push();

                break;

            }

            case 2:

            {

                pop();

                break;

            }

            case 3:

            {

                display();

                break;

            }

            case 4:

            {

                printf("\n\t EXIT POINT ");

                break;

            }

            default:

            {

                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

            }

      }

    }while(choice!=4);

    return 0;

}

void push()

{

    if(top>=n-1)

    {

        printf("\n\tSTACK is over flow");

    }

    else

    {

        printf(" Enter a value to be pushed:");

        scanf("%d",&x);

        top++;

        stack[top]=x;

    }

}

void pop()

{

    if(top<=-1)

    {

        printf("\n\t Stack is under flow");

    }

    else

    {

        printf("\n\t The popped elements is %d",stack[top]);

        top--;

    }

}

void display()

{

    if(top>=0)

    {

        printf("\n The elements in STACK \n");

        for(i=top; i>=0; i--)

            printf("\n%d",stack[i]);

        printf("\n Press Next Choice");

    }

    else

    {

        printf("\n The STACK is empty");

    }

}

**OUTPUT:**

