CSA0630

DESIGN ANALYSIS AND ALGORITHMS FOR SORRTING

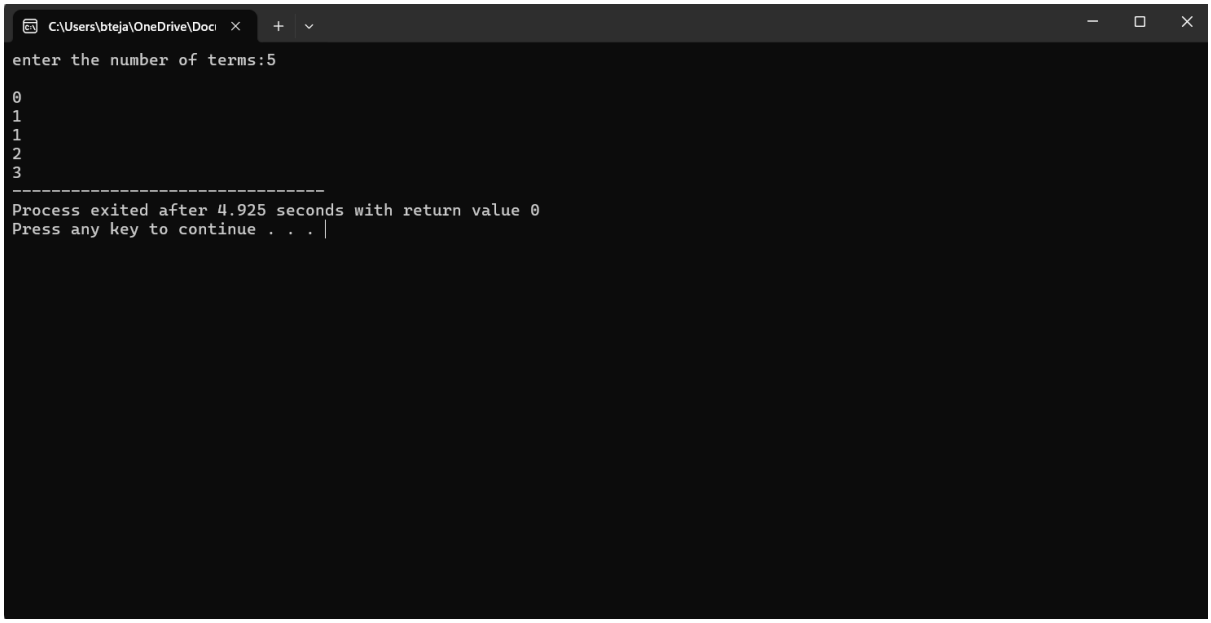
PRACTICAL SESSION DAY1

1. Write a program to Print Fibonacci Series using recursion

PROGRAM:

```
#include<stdio.h>
int fibanocci(int n)
{
        if(n<=1)
        return n;
        else
        return fibanocci(n-1)+ fibanocci(n-2);
}
int main()
{
        int n,i;
        printf("enter the number of terms:");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                printf("\n%d",fibanocci(i));
        }
        return 0;
}
```

OUTPUT:



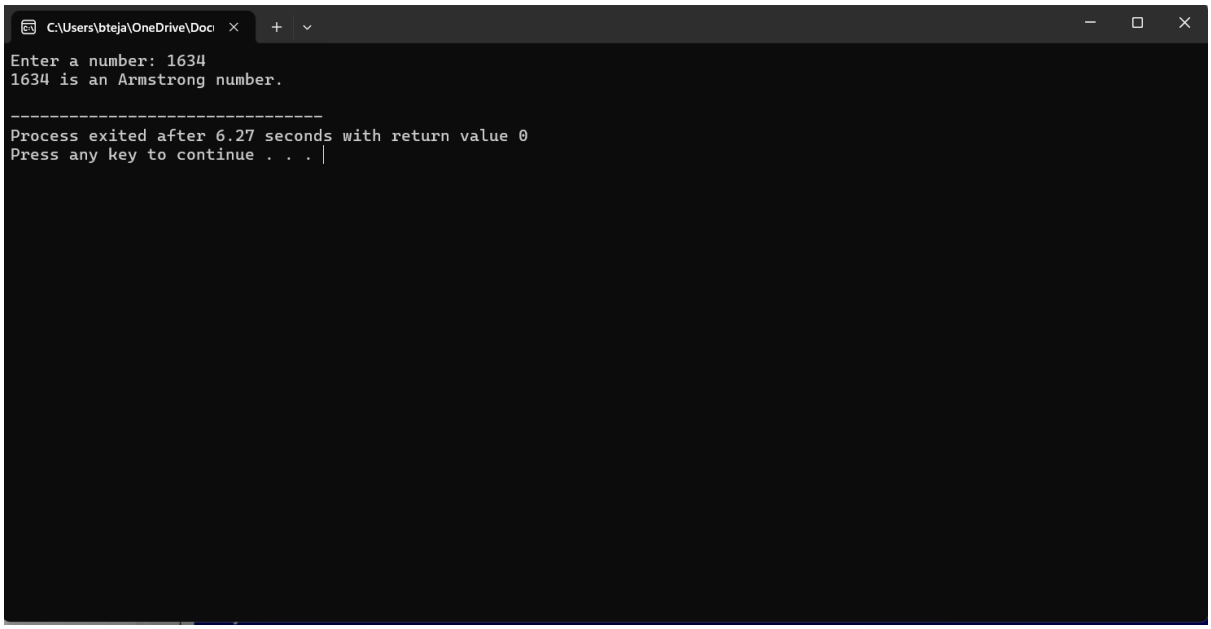
2. Write a program to check the given no is Armstrong or not using recursive function

PROGRAM:

```
#include <stdio.h>
#include <math.h>
int countDigits(int num) {
    if (num == 0) {
        return 0;
    } else {
        return 1 + countDigits(num / 10);
    }
}
int isArmstrong(int num, int n) {
    if (num == 0) {
        return 0;
    } else {
        return pow(num % 10, n) + isArmstrong(num / 10, n);
    }
}
int main() {
    int number, sum = 0, temp, numDigits;
    printf("Enter a number: ");
    scanf("%d", &number);
    numDigits = countDigits(number);
    sum = isArmstrong(number, numDigits);
    if (sum == number) {
        printf("%d is an Armstrong number.\n", number);
    } else {
        printf("%d is not an Armstrong number.\n", number);
    }
```

```
    return 0;
}
```

OUTPUT:

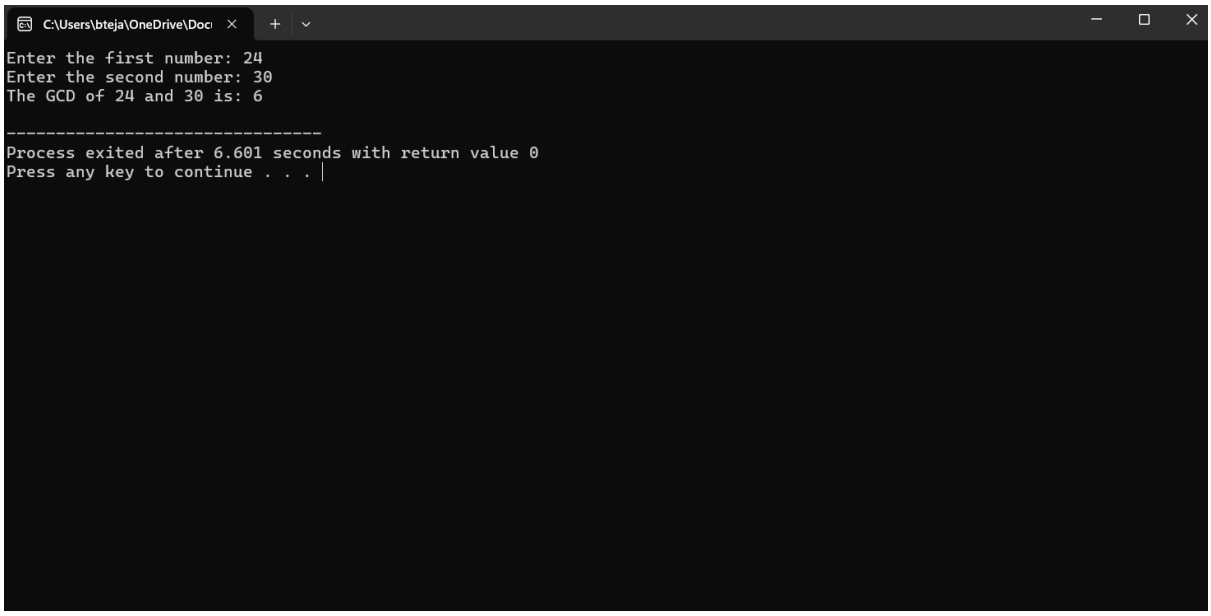


3.Write a program to find the GCD of two numbers using recursive factorisation

PROGRAM:

```
#include <stdio.h>
int find_gcd_recursive(int a, int b) {
    if (b == 0) {
        return a;
    }
    else {
        return find_gcd_recursive(b, a % b);
    }
}
int main() {
    int num1, num2;
    printf("Enter the first number: ");
    scanf("%d", &num1);
    printf("Enter the second number: ");
    scanf("%d", &num2);
    int gcd = find_gcd_recursive(num1, num2);
    printf("The GCD of %d and %d is: %d\n", num1, num2, gcd);
    return 0;
}
```

OUTPUT:

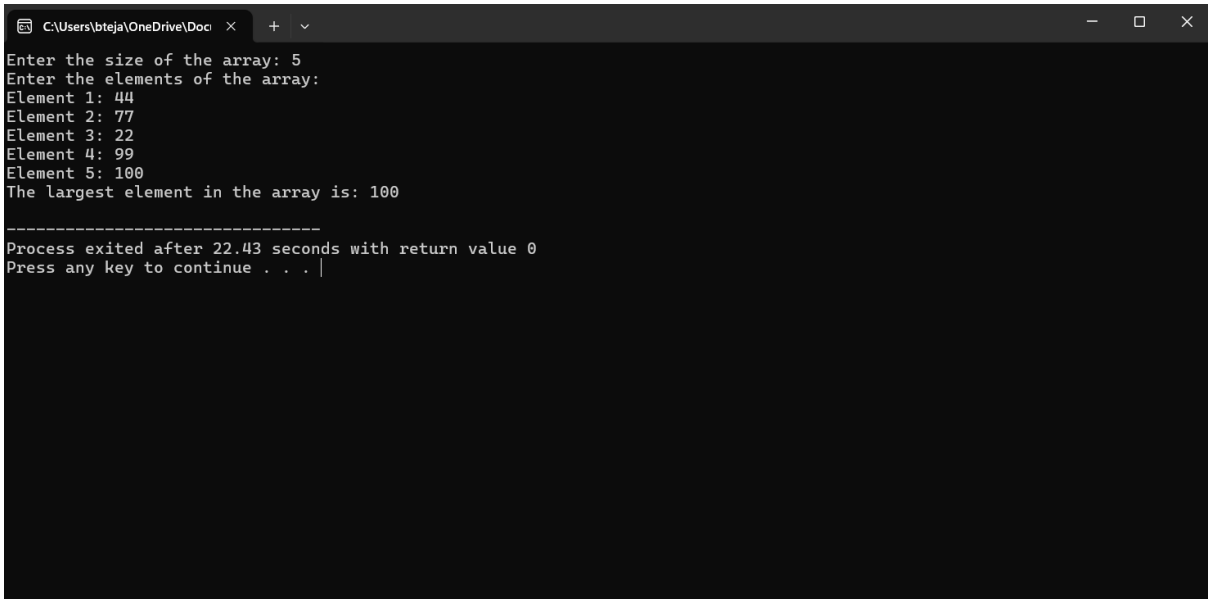


4.Write a program to get the largest element of an array.

PROGRAM:

```
#include <stdio.h>
int find_largest_element(int arr[], int size) {
    int largest = arr[0];
    for (int i = 1; i < size; ++i) {
        if (arr[i] > largest) {
            largest = arr[i];
        }
    }
    return largest;
}
int main() {
    int size;
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    int arr[size];
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; ++i) {
        printf("Element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
    int largest = find_largest_element(arr, size);
    printf("The largest element in the array is: %d\n", largest);
    return 0;
}
```

OUTPUT:

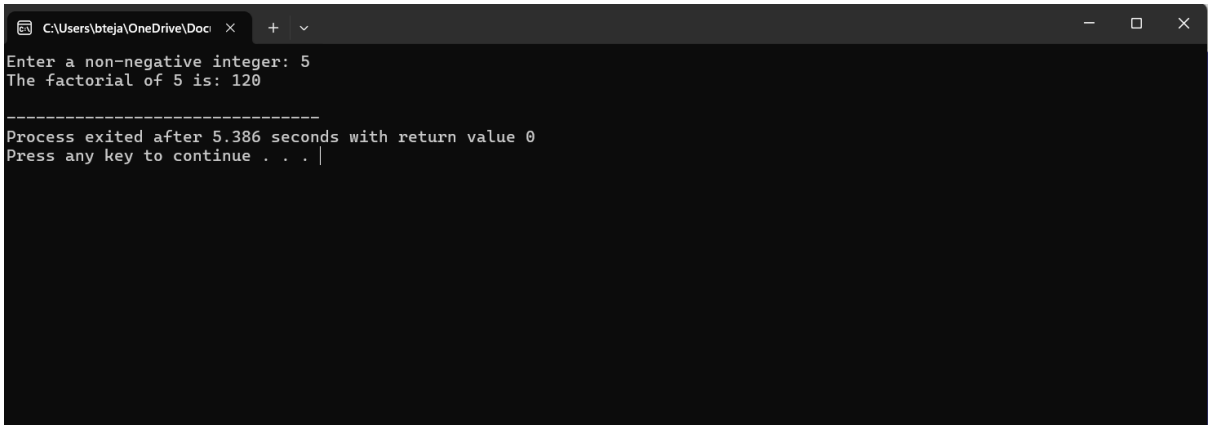


5.Write a program to find the Factorial of a number using recursion.

PROGRAM:

```
#include <stdio.h>
int factorial(int n)
 {
   if (n == 0 || n == 1)
           {
      return 1;
    }
           else
           {
      return n * factorial(n - 1);
    }
}
int main() {
   int num;
   printf("Enter a non-negative integer: ");
   scanf("%d", &num);
   if (num < 0) {
      printf("Factorial is not defined for negative numbers.\n");
   } else {
      int result = factorial(num);
      printf("The factorial of %d is: %d\n", num, result);
   }
   return 0;
}
```

OUTPUT:

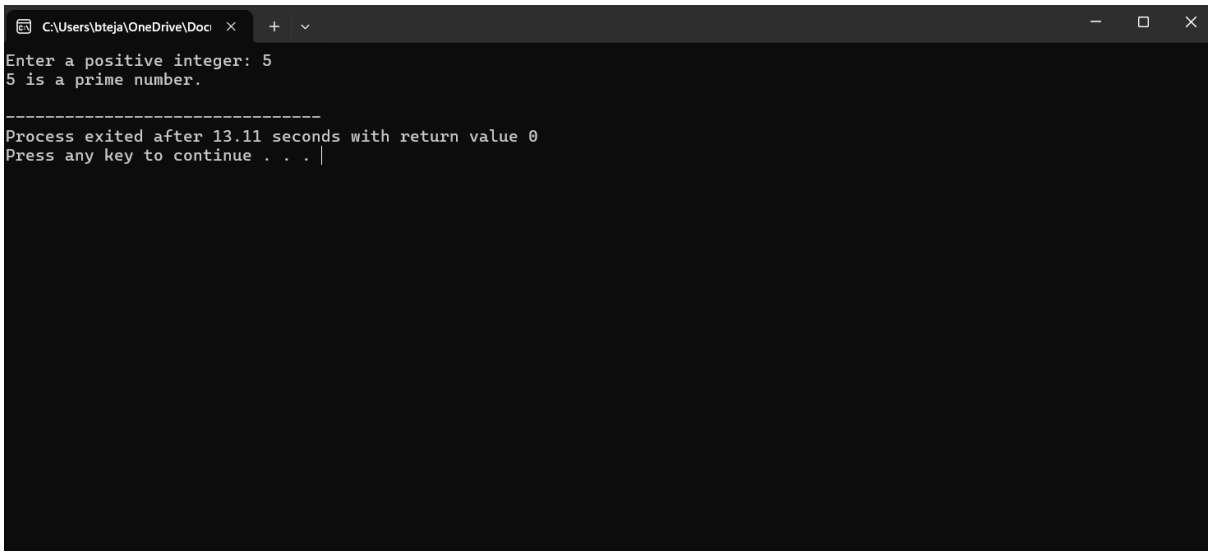


6.Write a program to check a number is a prime number or not using recursion

PROGRAM:

```
#include <stdio.h>
int is_prime_recursive(int num, int divisor) {
    if (num <= 1) {
        return 0;
    }
    if (divisor == 1) {
        return 1;
    }
    if (num % divisor == 0) {
        return 0;
    } else {
        return is_prime_recursive(num, divisor - 1);
    }
}
int main() {
    int num;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    if (is_prime_recursive(num, num / 2)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }
    return 0;
}
```

OUTPUT:

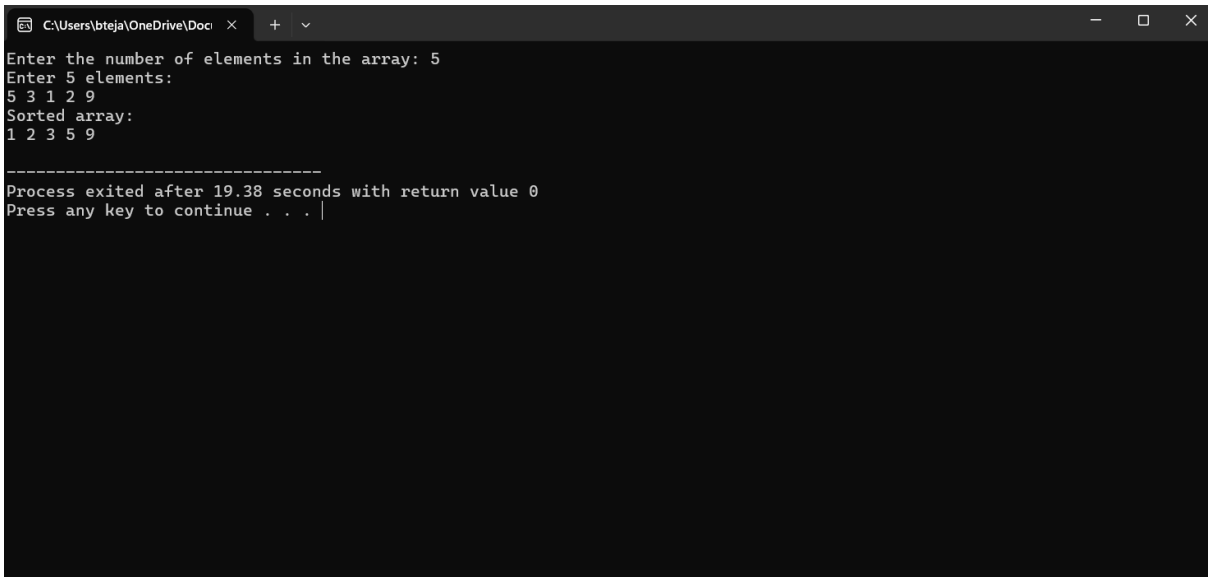


7.Write a program to perform Selection sort

PROGRAM:

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int i, j, minIndex, temp;
    for (i = 0; i < n - 1; i++) {
        minIndex = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
          temp = arr[minIndex];
        arr[minIndex] = arr[i];
        arr[i] = temp;
    }
    printf("Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

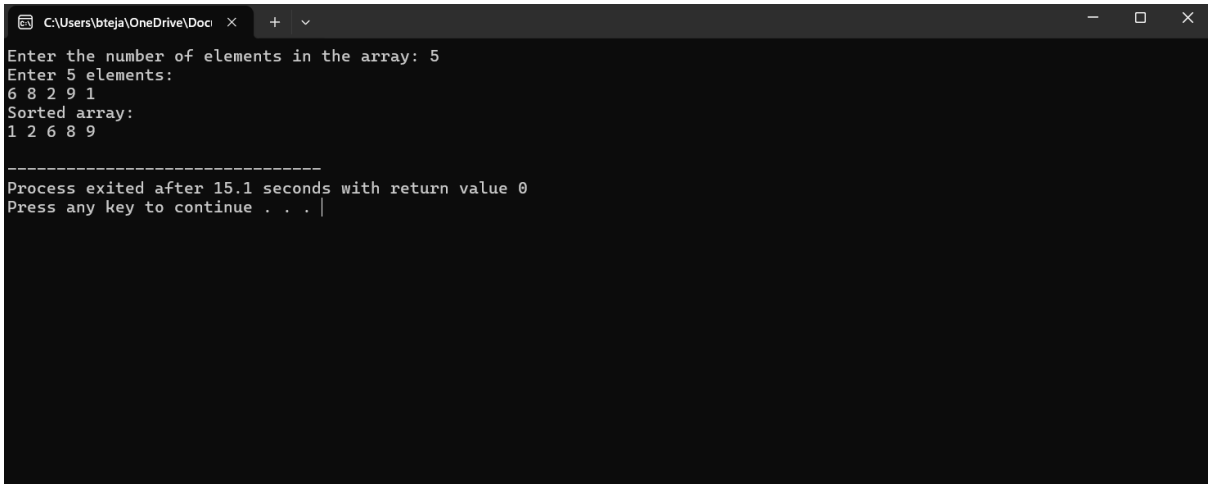


8.Write a program to perform Bubble sort

PROGRAM:

```
#include <stdio.h>
int main() {
    int n;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {

                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
printf("Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:



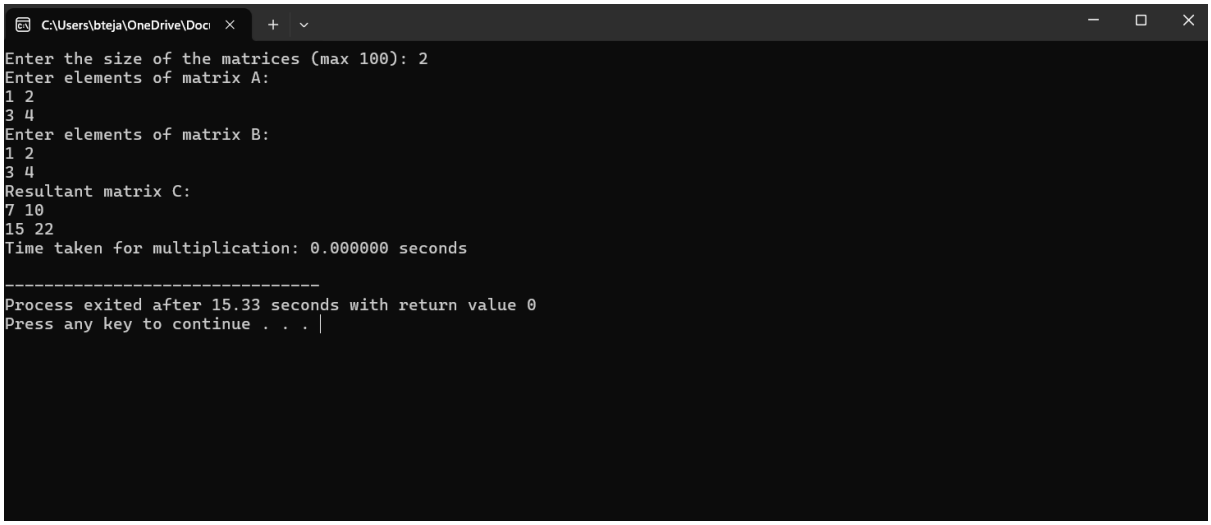
9.Write a program for calculating time complexity for multiply two Matrix

PROGRAM:

```
#include <stdio.h>
#include <time.h>
#define N 100
int main()
{
    int A[N][N], B[N][N], C[N][N];
    int size;
    printf("Enter the size of the matrices (max %d): ", N);
    scanf("%d", &size);
    printf("Enter elements of matrix A:\n");
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            scanf("%d", &A[i][j]);
        }
    }
    printf("Enter elements of matrix B:\n");
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            scanf("%d", &B[i][j]);
        }
    }
    clock_t start = clock();
   for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            C[i][j] = 0;
            for (int k = 0; k < size; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    clock_t end = clock();

    printf("Resultant matrix C:\n");
```

```
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("%d ", C[i][j]);
        }
        printf("\n");
    }

    double time_taken = ((double)(end - start)) / CLOCKS_PER_SEC;
    printf("Time taken for multiplication: %f seconds\n", time_taken);

    return 0;
}
```

OUTPUT:



10.Write a program for to check whether a given String is Palindrome or not using recursion

PROGRAM:

```
#include <stdio.h>
#include <string.h>
int isPalindrome(char str[], int start, int end)
 {
    if (start >= end)
            {
        return 1;
    }
    if (str[start] == str[end])
             {
        return isPalindrome(str, start + 1, end - 1);
     } else {
        return 0;
    }
}
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    if (isPalindrome(str, 0, strlen(str) - 1)) {
```

```
        printf("%s is a palindrome.\n", str);
    } else {
        printf("%s is not a palindrome.\n", str);
    }
    return 0;
}
```

OUTPUT: