

**Assignment-1**

**Name: - Mann Kothari**

**Reg. No.: - 22BCB7064**

**Course Title: - Design and Analysis of Algorithm (Embedded Lab)**

**Course Code: - CSE3023**

**Slot: - L21+L22**

**Submitted to: - Prof. Tanikella Divya Naga Pavani**

* **INDEX**

**Q1.** Find the Sum of two Positive Integers.

**Q2.** Find the Multiplication of two Positive Integers.

**Q3** Implement N terms of the Fibonacci series using recursion and find out the number of functions calls for different values of n.

**Q4** Find the GCD of two numbers.

**Q5** Find the factorial of a given number.

**Q6** Towers of Hanoi.

**Q7** Permutation generator

input: {a.b.c}

output: a,b,c; a,c,b; b,a,c; b,c,a; c,a,b; c,b,a;

all permutations without repetition.

**Q8** Given a set of characters and a positive integer k, print all possible strings of length k that can be formed from the given set.

{I/P: a.b. K=3, O/P: aaa, aab, abb, aba, …}

* Implement the following problems using recursion in C/C++.

1. Find the Sum of two Positive Integers.

#include<stdio.h>

void main()

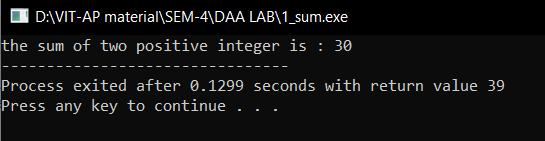
{

    int a=10, b=20;

    printf("the sum of two positive integer is : %d",a+b);

}

* **Output:**

****

1. Find the Multiplication of two Positive Integers.

#include<stdio.h>

void main()

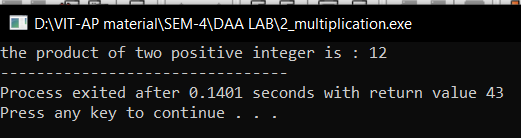
{

    int a=2, b=6;

    printf("the product of two positive integer is : %d",a\*b);

}

* **Output:**



1. Implement N terms of the Fibonacci series using recursion and find out the number of functions calls for different values of n.

Algorithm: -

Algorithm fibonacci (n: nonnegative integer)

if n = 0 then return 0;

else

x: = 0;

y: = 1;

for i: =1 to n -1

z: = x + y;

x: = y;

y: = z;

fibonacci(n-1);

{output is the nth Fibonacci number}

Code: -

#include<stdio.h>

void fibonacci(int n, int \*calls){

    (\*calls)++;

    static int n1=0,n2=1,n3;

    if(n>0){

         n3 = n1 + n2;

         n1 = n2;

         n2 = n3;

         printf("%d ",n3);

         fibonacci(n-1, calls);

    }

}

int main()

{

     int n;

     int n1=0, n2=1;

     int calls=0;

     printf("enter the value of n: ");

     scanf("%d", &n);

     printf("%d %d\t", n1,n2);

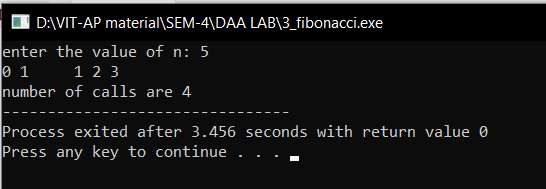
     fibonacci(n-2, &calls);

     printf("\nnumber of calls are %d", calls);

     return 0;

}

* **Output:**



1. Find the GCD of two numbers.

#include <stdio.h>

#include <math.h>

int gcd(int a, int b)

{

    int result = ((a < b) ? a : b);

    while (result > 0) {

        if (a % result == 0 && b % result == 0) {

            break;

        }

        result--;

    }

    return result;

}

int main()

{

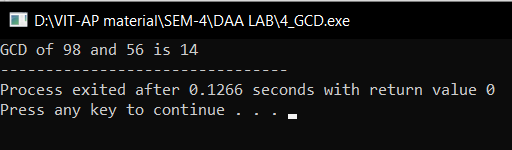
    int a = 98, b = 56;

    printf("GCD of %d and %d is %d ", a, b, gcd(a, b));

    return 0;

}

* **Output:**



1. Find the factorial of a given number.

Algorithm: -

Algorithm factorial(n)

{

res: = 1;

for i: = n to 0 step-1 do

{

res: = res\*I;

}

return res;

Code: -

#include <stdio.h>

int factorial(int n)

{

    int result = 1, i;

    for (i = n; i > 0; i--) {

        result \*= i;

    }

    return result;

}

int main()

{

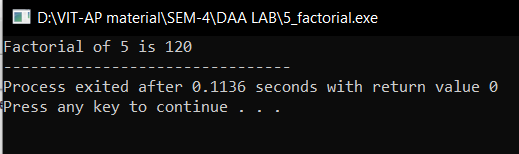
    int num = 5;

    printf("Factorial of %d is %d", num, factorial(num));

    return 0;

}

* **Output:**



1. Towers of Hanoi.

Algorithm: -

Algorithm TowerOfHanoi(n, from\_rod, to\_rod, aux\_rod)

{

if (n= =1)then

{

write(“ Move disk 1 from rod %c to rod %c”);

return;

}

TowerOfHanoi(n-1, from\_rod, aux\_rod, to\_rod);

Write(“Move disk %d from rod %c to rod %c”);

TowerOfHanoi((n-1, aux\_rod, to\_rod, from\_rod);

}

Code: -

#include <stdio.h>

void TowerOfHanoi(int n, char from\_rod, char to\_rod, char aux\_rod)

{

    if (n == 1)

    {

        printf("\n Move disk 1 from rod %c to rod %c", from\_rod, to\_rod);

        return;

    }

    TowerOfHanoi(n-1, from\_rod, aux\_rod, to\_rod);

    printf("\n Move disk %d from rod %c to rod %c", n, from\_rod, to\_rod);

    TowerOfHanoi(n-1, aux\_rod, to\_rod, from\_rod);

}

int main()

{

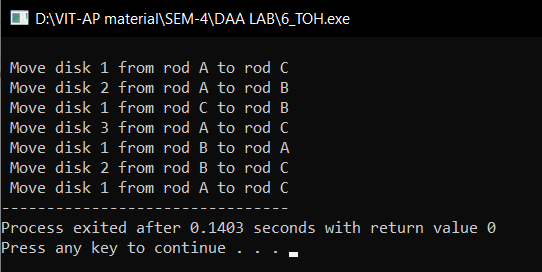
    int n = 3;

    TowerOfHanoi(n, 'A', 'C', 'B');

    return 0;

}

* **Output:**



1. Permutation generator

input: {a.b.c}

output: a,b,c; a,c,b; b,a,c; b,c,a; c,a,b; c,b,a;

all permutations without repetition.

#include <stdio.h>

#include <string.h>

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

{

    char temp;

    temp = \*x;

    \*x = \*y;

    \*y = temp;

}

void permute(char\* a, int l, int r)

{

    int i;

    if (l == r)

        printf("%s\n", a);

    else {

        for (i = l; i <= r; i++) {

            swap((a + l), (a + i));

            permute(a, l + 1, r);

            swap((a + l), (a + i));

        }

    }

}

int main()

{

    char str[] = "ABC";

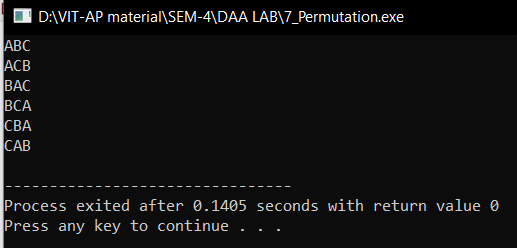
    int n = strlen(str);

    permute(str, 0, n - 1);

    return 0;

}

* **Output:**



1. Given a set of characters and a positive integer k, print all possible strings of length k that can be formed from the given set.

{I/P: a.b. K=3, O/P: aaa, aab, abb, aba, …}

#include <stdio.h>

#include <string.h>

void generateStrings(const char \*characters, int k, char \*current)

{

    if (k == 0)

    {

        printf("%s\n", current);

        return;

    }

    for (int i = 0; i < strlen(characters); i++)

    {

        current[k - 1] = characters[i];

        generateStrings(characters, k - 1, current);

    }

}

int main()

{

    const char \*input\_characters = "abc"; // replace with your set of characters

    int k = 3;                      // replace with your desired length

    char current[k + 1];

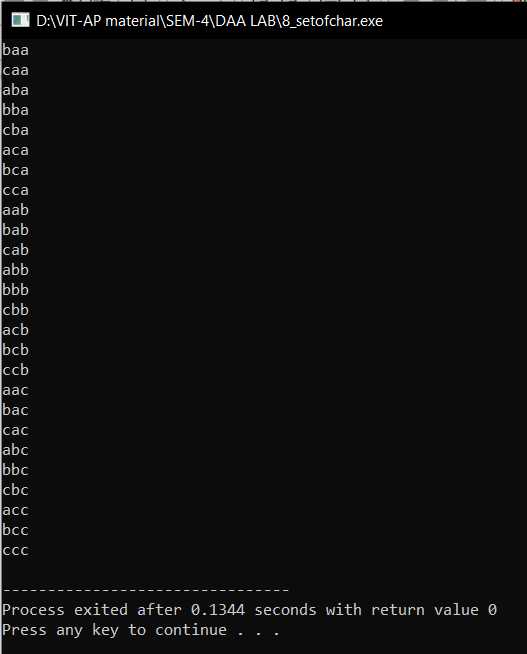
    current[k] = '\0'; // null-terminate the string

    generateStrings(input\_characters, k, current);

    return 0;

}

* **Output:**



|  |  |  |
| --- | --- | --- |
| Algorithm | Time Complexity | Space Complexity |
| Sum | O(1) | O(1) |
| Multiplication | O(1) | O(1) |
| Fibonacci | O(n) | O(n) |
| GCD | O(n) | O(1) |
| Factorial | O(n) | O(1) |
| Permutation Generator | O(2n) | O(2n) |
| Set of Characters | O(n!) | O(n!) |