1. **Write a program to Print Fibonacci Series using recursion.**

#include<stdio.h>

int fib(int n)

{

if(n==0){

return 0;

}

else if (n==1)

{

return 1;

}

else{ **TIME COMPLEXITY:**

return(fib(n-1)+fib(n-2)); O(2^n)

}

}

int main(){

int n;

scanf("%d",&n);

for (int i=0;i<10;i++)

{

printf(" %d, ",fib(i));

}

return 0;

}

**Output:**

A black screen with white text

Description automatically generated

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

#include <stdio.h>

#include <math.h>

int a(int num, int d) {

if (num == 0)

return 0;

else{

return pow(num % 10, d) + a(num / 10, d);

}

}

int main() { **TIME COMPLEXITY:**

int n=6, d= 0; O(log n)

int originalNum = n;

while (originalNum != 0) {

originalNum /= 10;

++d;

}

if (a (n, d) == n)

printf("%d is an Armstrong number.\n", n);

else

printf("%d is not an Armstrong number.\n", n);

return 0;

}

**Output:**

A screen shot of a computer

Description automatically generated

1. **Write a program to find the GCD of two numbers using recursive factorization.**

#include <stdio.h>

int gcd(int n1, int n2);

int main() {

int n1, n2;

printf("Enter two positive integers: ");

scanf("%d %d", &n1, &n2);

printf("G.C.D of %d and %d is %d.", n1, n2, gcd(n1, n2));

return 0;

}

int gcd(int n1, int n2) { **TIME COMPLEXITY:**

if (n2 != 0) O(n)

return gcd(n2, n1 % n2);

else

return n1;

}

**Output:**

A screen shot of a computer

Description automatically generated

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

#include <stdio.h>

int main() {

int arr[]={23,87,90,0,65};

int n=sizeof(arr)/sizeof(arr[0]); **TIME COMPLEXITY:**

for (int i = 1; i < n; ++i) { O(n)

if (arr[0] < arr[i]) {

arr[0] = arr[i];

}

}

printf("Largest element = %d", arr[0]);

return 0;

}

**Output:**



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

#include<stdio.h>

int fact(int n){

if(n<=1){

return 1;

}

else{

return(n\*fact(n-1)); **TIME COMPLEXITY:**

} O(n)

}

int main(){

int n;

scanf("%d",&n);

printf("%d",fact(n));

return 0;

}

**Output:**

A black screen with white text

Description automatically generated

1. **Write a program for to copy one string to another using recursion**.

#include <stdio.h>

void copy(char [], char [], int);

int main()

{

char str1[20], str2[20];

printf("Enter string to copy: "); **TIME COMPLEXITY:**

scanf("%[^\n]s", str1); O(n)

copy(str1, str2, 0);

printf("Copying success.\n");

printf("The first string is: %s\n", str1);

printf("The second string is: %s\n", str2);

return 0;

}

void copy(char str1[], char str2[], int index)

{

str2[index] = str1[index];

if (str1[index] == '\0')

return;

copy(str1, str2, index + 1);

}

**Output:**

A computer screen with white text

Description automatically generated

**7. Write a program to print the reverse of a string using recursion.**

# include <stdio.h>

void reverse(char \*str)

{

if (\*str)

{

reverse(str + 1); **TIME COMPLEXITY:**

printf("%c", \*str); O(n)

}

}

int main()

{

char a[] = "tree";

reverse(a);

return 0;

}

**Output:**

A black screen with white text

Description automatically generated

**8. Write a program to generate all the prime numbers using recursion.**

#include<stdio.h>

#include<math.h>

int CheckPrime(int i,int num)

{

if(num==i)

return 0;

else

if(num%i==0) **TIME COMPLEXITY:**

return 1; O(n^2)

else{

return CheckPrime(i+1,num);

}

}

int main()

{

int n,i;

scanf("%d",&n);

printf("Prime Number Between 1 to n are: ");

for(i=2;i<=n;i++){

if(CheckPrime(2,i)==0){

printf("%d ",i);

}

}

}

**Output:**

A black background with white text

Description automatically generated

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

#include<stdio.h>

int CheckPrime(int i,int num)

{

if(num==i)

return 0;

else

if(num%i==0)

return 1;

else{ **TIME COMPLEXITY:**

return CheckPrime(i+1,num); O(√n)

}

}

int main()

{

int num = 11;

if(CheckPrime(2,num)==0)

printf("It is a Prime Number.");

else

printf("It is not a Prime Number.");

}

**Output:**

A black screen with white text

Description automatically generated

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

#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 0;

return isPalindrome(str, start+1, end-1);

}

int main() {

char str[100];

printf("Enter a string: ");

fgets(str, sizeof(str), stdin); **TIME COMPLEXITY:**

int len = strlen(str) - 2; O(n)

if(isPalindrome(str, 0, len))

printf("%s is a Palindrome.\n", str);

else

printf("%s is not a Palindrome.\n", str);

return 0;

}

**Output:**

A screen shot of a computer

Description automatically generated