DS Lab

Programs with Recursion

**Implement binary search, factorial and tower of hanoi using recursion**

**Theory**

Recursion is the process of repeating items in a self-similar way. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function.

**Binary search:**

Binary Search is a searching algorithm for finding an element's position in a sorted array. In this approach, the element is always searched in the middle of a portion of an array.

**Tower of Hanoi:**

Tower of Hanoi is a mathematical puzzle where we have three rods (A, B, and C) and N disks. Initially, all the disks are stacked in decreasing value of diameter i.e., the smallest disk is placed on the top and they are on rod A. The objective of the puzzle is to move the entire stack to another rod (here considered C), obeying the following simple rules:

Only one disk can be moved at a time.

Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

No disk may be placed on top of a smaller disk

1. Factorial

Pseudocode

int factorial(n)

if n==1

return 1

else

return n\*factorial(n-1)

Trace

factorial(5) 🡪 5\*factorial(4) 🡪 5\*4\*factorial(3) 🡪 5\*4\*3\*factorial(2)

* 5\*4\*3\*2\*factorial(1) 🡪 5\*4\*3\*2\*1 = 120

**Code**

#include<stdio.h> //include standard header file

int fact(int n) //factorial recursive function

{

if(n==1) //factorial of 1 is 1

return 1;

else

return (n\*fact(n-1)); //multiply n by factorial of n-1

}

int main()

{

int n; //initialize n

printf("Enter number: ");

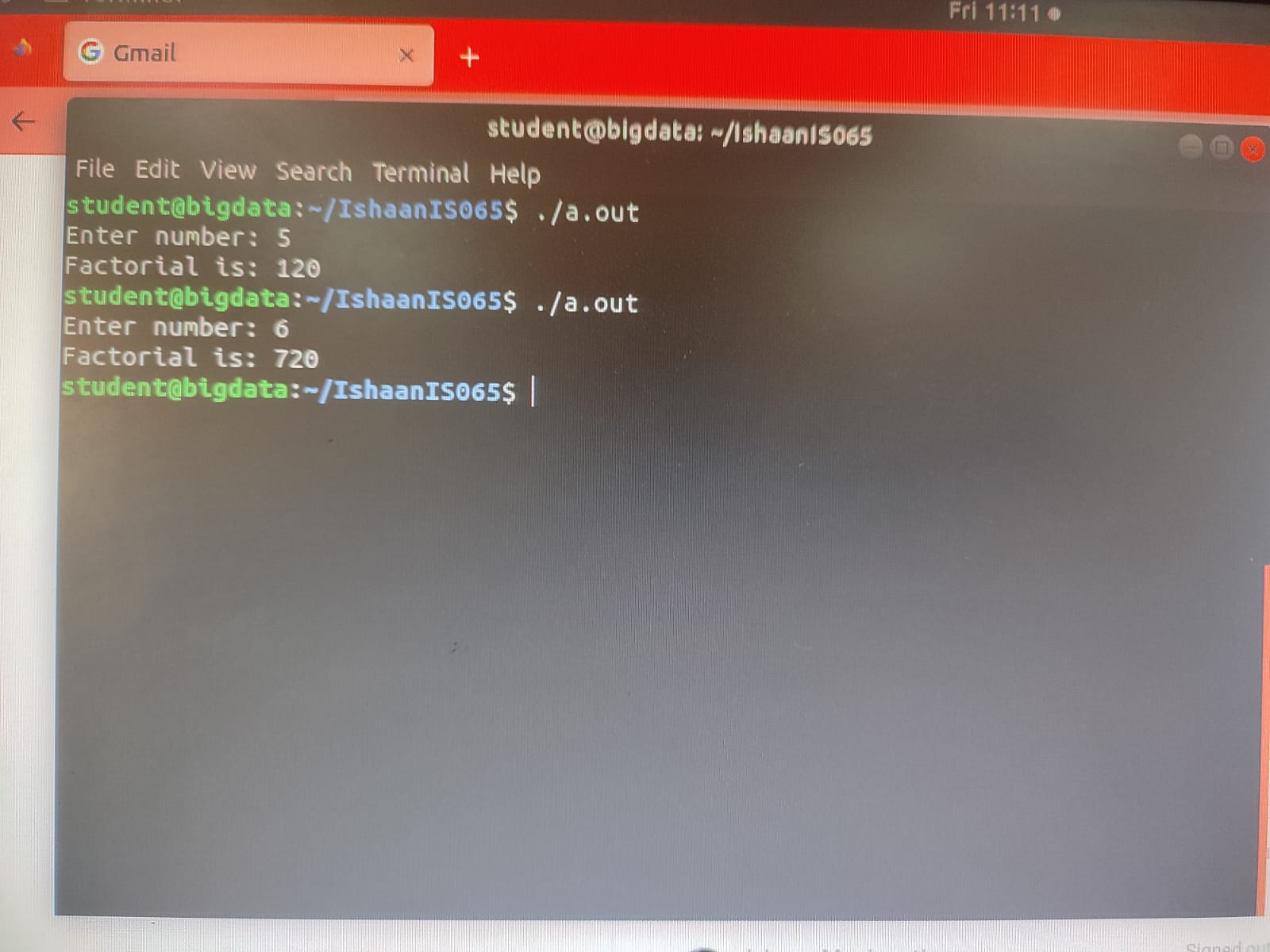
scanf("%d", &n); //store n value by user

printf("Factorial is: %d\n", fact(n)); //print factorial value

return 0;

}

**Output**

****

1. Binary Search

Pseudocode

Int binarySearch(array, n, low, high)

mid=low+high/2

for(i=0; i<=high; i++)

if array[mid]==n

return mid;

else if array[mid]>n

return binarySearch(array, n, low, mid-1)

else if array[mid]<n

return binarySearch(array, n, mid+1, high)

else

return -1

**Code**

#include<stdio.h> //include standard header file

int binarySearch(int arr[], int x, int low, int high) //binary //search recursive function

{

int mid=(high+low)/2; //initializing mid to mid index //of array

if(arr[mid]==x) //if element found at mid index of //array

return mid;

else if(arr[mid]>x) //search left side of mid

return binarySearch(arr, x, low, mid-1);

else if(arr[mid]<x) //search right side of mid

return binarySearch(arr, x, mid+1, high);

else

return -1; //if element not found in array

}

int main()

{

int array[10], x;

int n= sizeof(array)/sizeof(array[0]); //length of array

printf("Enter array elements in order (Max elements=10): \n"); //ask for user defined array

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

{

scanf("%d", &array[i]); //store values in order //in array

}

printf("Enter element to be searched: "); //user input

scanf("%d", &x); //store value in x

int result=binarySearch(array, x, 0, n-1); //search x //in array

if(result==-1)

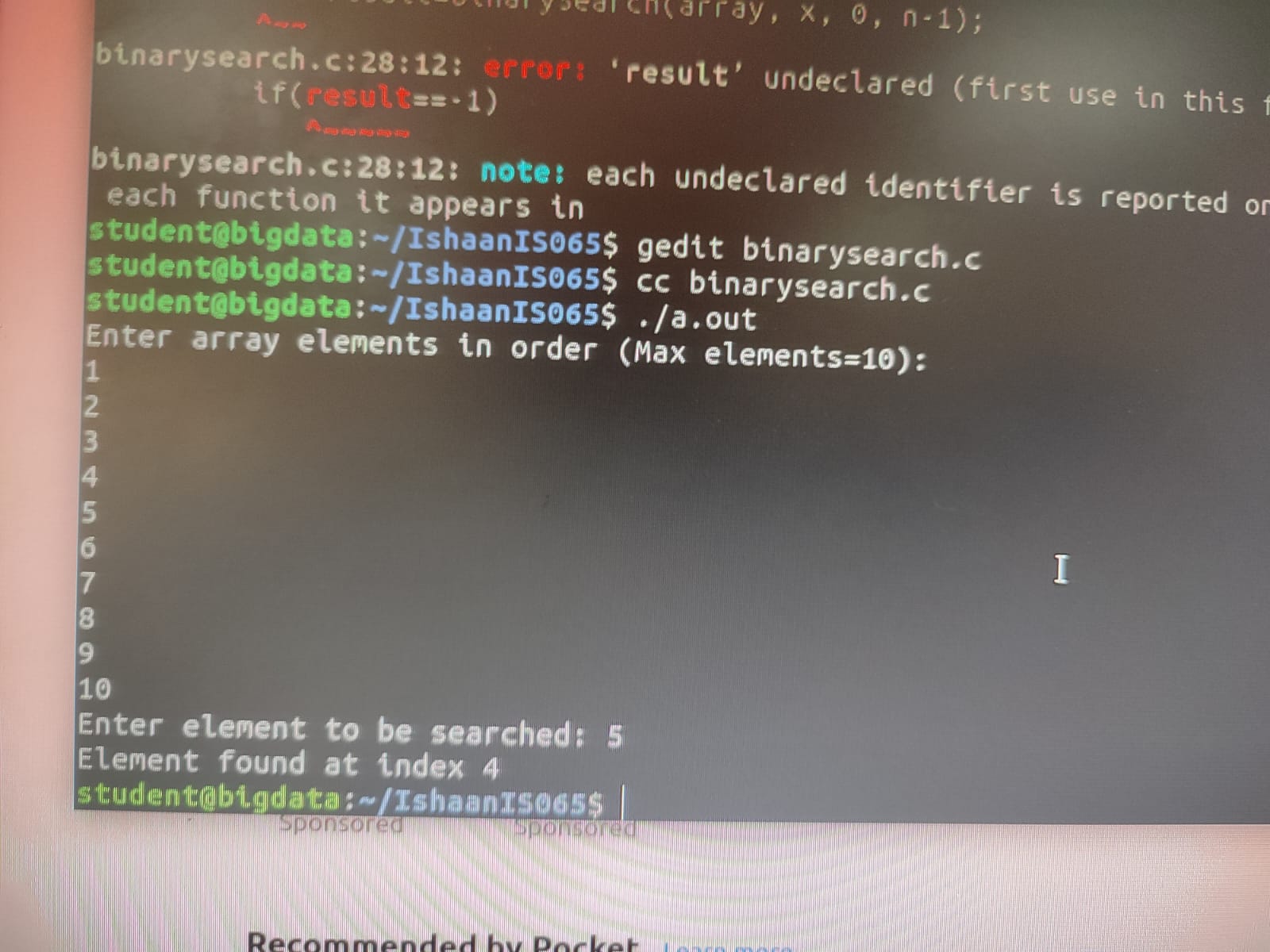
printf("Not found\n");

else

printf("Element found at index %d\n", result);

}

**Output**

****

1. Tower of Hanoi

Pseudocode

Void towerOfHanoi(n, from\_rod, to\_rod, aux\_rod)

If n==1

print – move disk 1 from (from\_rod) to (to\_rod)

towerOfHanoi(n-1, from\_rod, aux\_rod, to\_rod)

print – move disk n rom (from\_rod) to (to\_rod)

towerOfHanoi(n-1, aux\_rod, to\_rod, from\_rod)

**Code**

#include<stdio.h> //include standard header file

void towerOfHanoi(int n, char from\_rod, char to\_rod, char aux\_rod) //tower of hanoi recursive function

{

if(n==1) //for the smallest disk on top

{

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

return;

}

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

printf("\nMove 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;

printf("Enter number of disks: "); //user input

scanf("%d", &n); //store in n

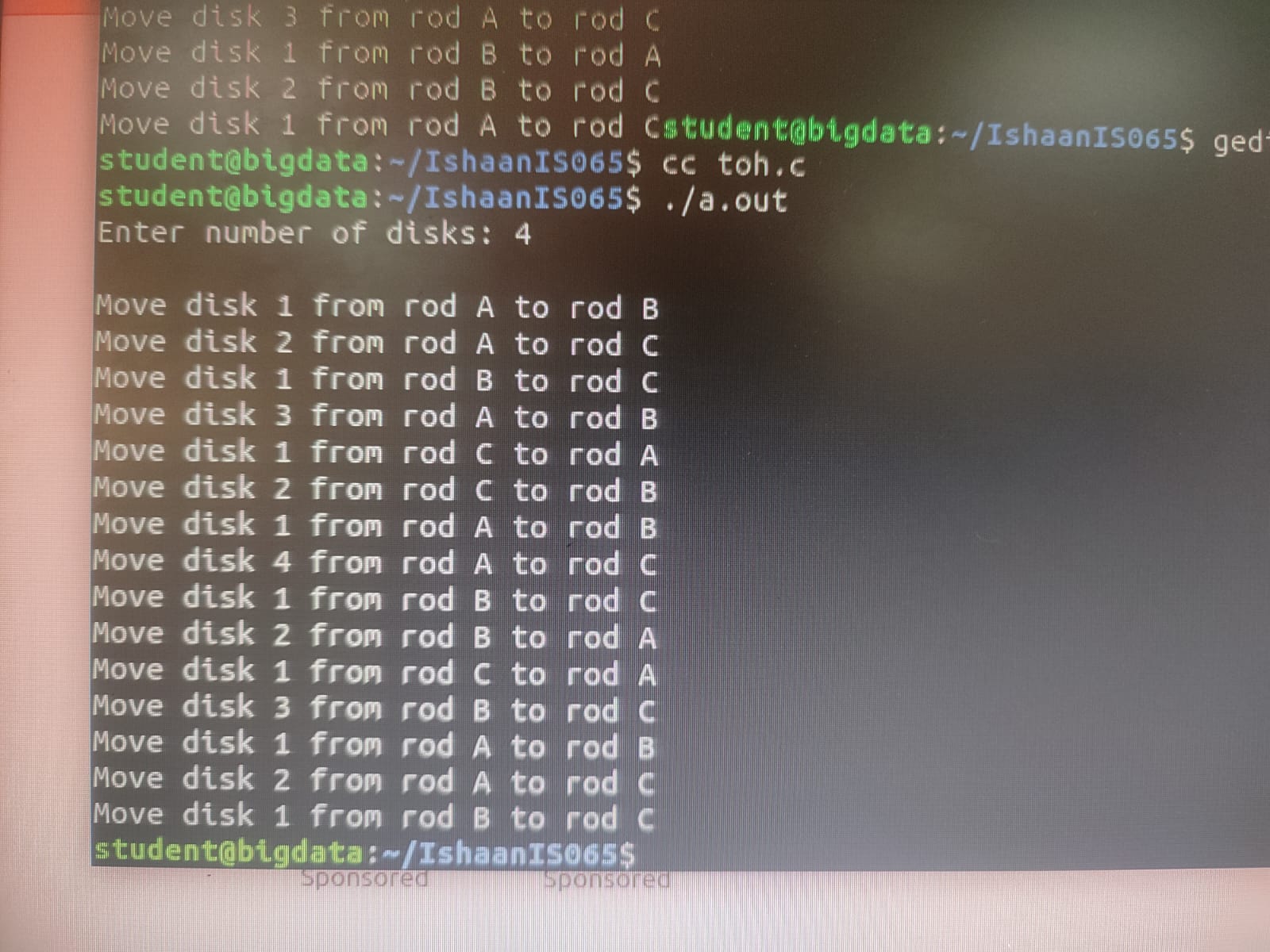
towerOfHanoi(n, 'A', 'C', 'B'); //A, B and C are rod //names

printf("\n");

return 0;

}

**Output**

****