Jesadaporn Pannok No.66070503410

**Lab 8 Lab Recursive Functions**

The Date of the Year

Example: Write a function in order to check for the validity of date as and input and repeat receiving the date until the given date is valid.

Input: 29/2/2023

Output: >> Input date is invalid. Enter a valid date again. Input: 29/2/2024

Output: >> Input date is valid.

*#include* <stdio.h>

*#define* *TRUE* 1

*#define* *FALSE* 0

*int* *isLeapYear*(*int* year) {

*if* ((year *%* 4 *==* 0 *&&* year *%* 100 *!=* 0) *||* (year *%* 400 *==* 0)) {

*return* *TRUE*;

} *else* {

*return* *FALSE*;

}

}

//

*int* *isValidDate*(*int* date, *int* month, *int* year) {

*int* *daysInMonth[]* *=* {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

*if* (*isLeapYear*(year)) {

*daysInMonth*[1]*++*; *// Increment the number of days for February in a leap year*

}

*if* (month *>=* 1 *&&* month *<=* 12) {

*if* (date *>=* 1 *&&* date *<=* *daysInMonth*[month *-* 1]) {

*return* *TRUE*;

}

}

*return* *FALSE*;

}

*void* *inputDate*() {

*int* *date*, *month*, *year*;

*printf*("Enter the date (DD/MM/YYYY): ");

*scanf*("%d/%d/%d", *&date*, *&month*, *&year*);

*if* (*isValidDate*(*date*, *month*, *year*)) {

*printf*(">> Input date is valid.\n");

} *else* {

*printf*(">> Input date is invalid. Enter a valid date again.\n");

*inputDate*();

}

}

*int* *main*() {

*inputDate*();

*return* 0;

}

Output

A screen shot of a computer

Description automatically generated

Summation

*#include* <stdio.h>

*int* *Summation*(*int* form, *int* to){

*if* (form *==* to)

{

*return* form; *// JEDSADAPORN PANNOK NO.66070503410*

}*else*{

*return* to *+* *Summation*(form , to*-*1);

}

}

*int* *main*(*void*) {

*int* *begin*,*end*;

*scanf*("%d\n%d",*&begin*,*&end*);

*printf*("summation form %d to %d is %d", *begin*, *end*, *Summation*(*begin*,*end*));

}

Output

A screen shot of a computer

Description automatically generated

Factorial

We will learn recursive functions from the factorial problem.

*#include* <stdio.h>

*int* *factorial*(*int* form, *int* to){

*if* (form *==* to)

{

*return* form;

}*else*{

*return* to *\** *factorial*(form , to*-*1);

}

}

*int* *main*(*void*) {

*int* *begin*,*end*; *// JEDSADAPORN PANNOK NO.66070503410*

*scanf*("%d\n%d",*&begin*,*&end*);

*printf*("factorial form %d to %d is %d", *begin*, *end*, *factorial*(*begin*,*end*));

}

Output

A screen shot of a computer code

Description automatically generated

Fibonacci Sequence

The most famous formulas in mathematics are the Fibonacci sequence. Each number in the sequence is the sum of the two numbers that precede it. Therefore, sequence looks like: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, and so on.

*#include* <stdio.h>

*int* *Fibonacci*(*int* n){

*if* (n *==* 0){

*return* 0;

}

*else* *if*(n *==* 1){

*return* 1; *// JEDSADAPORN PANNOK NO.66070503410*

}*else*{

*return* *Fibonacci*(n*-*1) *+* *Fibonacci*(n*-*2);

}

}

*int* *main*(*void*) {

*int* *n*;

*scanf*("%d",*&n*);

*printf*("Fibonacci = %d",*Fibonacci*(*n*));

}

Output

A screen shot of a computer

Description automatically generated

Sum of Digits of a Number

It is used to find the sum of digits of a number using recursion.

*#include* <stdio.h>

*int* *SumDigits*(*int* n){

*if* (n *==* 0){

*return* 0;

}*else*{

*return* (n*%*10) *+* *SumDigits*(n*/*10);

}

}

*int* *main*(*void*) {

*int* *n*;

*scanf*("%d",*&n*); *// JEDSADAPORN PANNOK NO.66070503410*

*printf*("SumDigits = %d",*SumDigits*(*n*));

}

Output

A computer screen shot of white text

Description automatically generated

Power of a Number

The product of multiplying a number by itself is called Power. Usually, with a Base number and an Exponent, the Power is expressed. The exponent refers to a small number written above and to the right of the base number. It depicts how many times the base number is multiplied.

*#include* <stdio.h>

*int* *Power*(*int* n,*int* topwr){

*if* (topwr *==* 0)

{

*return* 1;

}*else*{

*return* n *\** *Power*(n,topwr *-* 1); *// JEDSADAPORN PANNOK NO.66070503410*

}

}

*int* *main*(*void*) {

*int* *n*,*topwr*;

*scanf*("%d\n%d",*&n*,*&topwr*);

*printf*("Power\_of\_a\_Number = %d",*Power*(*n*,*topwr*));

}

Output

A screen shot of a computer screen

Description automatically generated

Least Common Multiple(LCM) of 2 Numbers

The least common multiple(LCM) of a number is the smallest number that is the product of two or more numbers.

*#include* <stdio.h>

*int* *LMC*(*int* n1,*int* n2){

*int* *sum*;

*sum* *=* n1*%*n2;

*if* (*sum* *==* 0)

{

*return* n1;

}

*return* n1 *\** *LMC*(n2,*sum*) */* *sum*;

}

*int* *main*(*void*) {*// JEDSADAPORN PANNOK NO.66070503410*

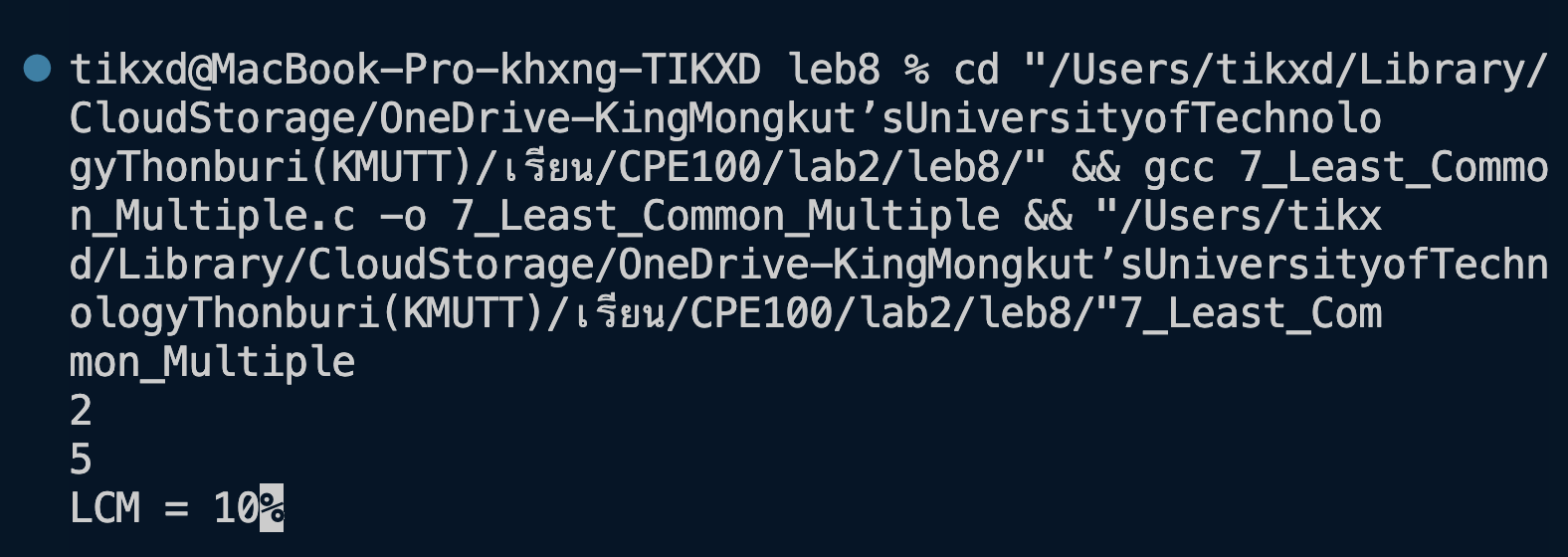
*int* *n1*,*n2*;

*scanf*("%d\n%d",*&n1*,*&n2*);

*printf*("LCM = %d",*LMC*(*n1*,*n2*));

}

Output



Greatest Common Divisor(GCD) of 2 Numbers

The most significant positive number is GCD of two or more integers that each of the integers is divisible, i.e., it is the most significant number that divides both of them. Thus, simply factorizing both numbers and multiplying common prime factors will give GCD.

*#include* <stdio.h>

*int* *GCD*(*int* n1, *int* n2) {

*int* *low*, *high*;

*if* (n1 *<* n2) {

*low* *=* n1;

*high* *=* n2;

} *else* {

*low* *=* n2;

*high* *=* n1;

}

*if* (*low* *==* 0) {

*return* *high*;

} *else* *if* (*low* *==* 1) {

*return* 1; *// JEDSADAPORN PANNOK NO.66070503410*

} *else* {

*return* *GCD*(*low*, *high* *%* *low*);

}

}

*int* *main*(*void*) {

*int* *n1*, *n2*;

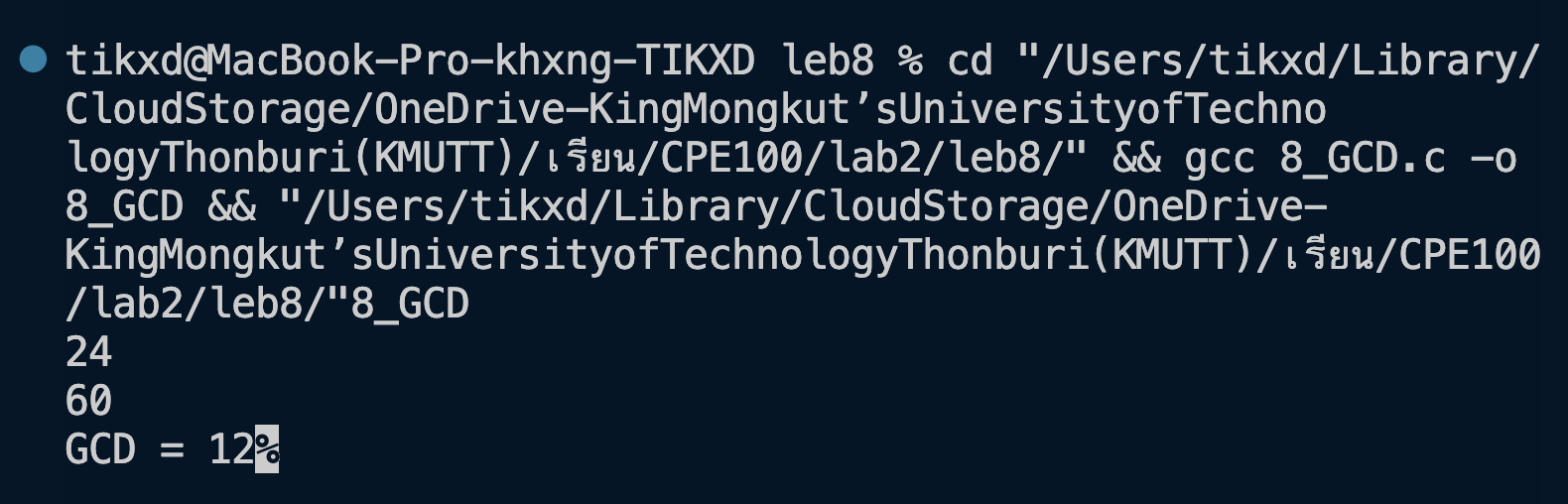
*scanf*("%d\n%d", *&n1*, *&n2*);

*printf*("GCD = %d", *GCD*(*n1*, *n2*));

*return* 0;

}

Output



Tower of Hanoi

A mathematical puzzle where we have three rods and n disks is known as the Tower of Hanoi. Here the puzzle is to move the entire stack to another rod, obeying the following simple rules: 1. At a time, one disk can move. 2. It works by taking the upper disk from one stack and placing it on top of another stack, i.e., a disk can move if it is the uppermost disk on a stack. 3. A larger disk can not be put on top of a smaller disk.

*#include* <stdio.h>

*void* *towerOfHanoi*(*int* n, *char* source, *char* auxiliary, *char* target) {

*if* (n *==* 1) {

*printf*("Move disk 1 from %c to %c\n", source, target);

*return*;

}

*towerOfHanoi*(n *-* 1, source, target, auxiliary);

*printf*("Move disk %d from %c to %c\n", n, source, target);

*towerOfHanoi*(n *-* 1, auxiliary, source, target);

}

*int* *main*() {*// JEDSADAPORN PANNOK NO.66070503410*

*int* *n*;

*printf*("Enter the number of disks: ");

*scanf*("%d", *&n*);

*towerOfHanoi*(*n*, 'A', 'B', 'C');

*return* 0;

}

Output

A screenshot of a computer program

Description automatically generated

Reverse a string

As per the title, it reverses the original string, i.e., modifying the original string and arranging it in a reverse manner, starting from the last character to the first character.

*#include* <stdio.h>

*#include* <string.h>

*void* *reverseString*(*char* *\**str) {

*if* (*strlen*(str) *==* 0) {

*return*;

} *else* {

*reverseString*(str *+* 1);

*printf*("%c", str[0]);

}

}

*int* *main*() {

*char* *input*[100];

*printf*("Enter your String: ");

*scanf*("%s", *input*);

*printf*("Switched text: ");

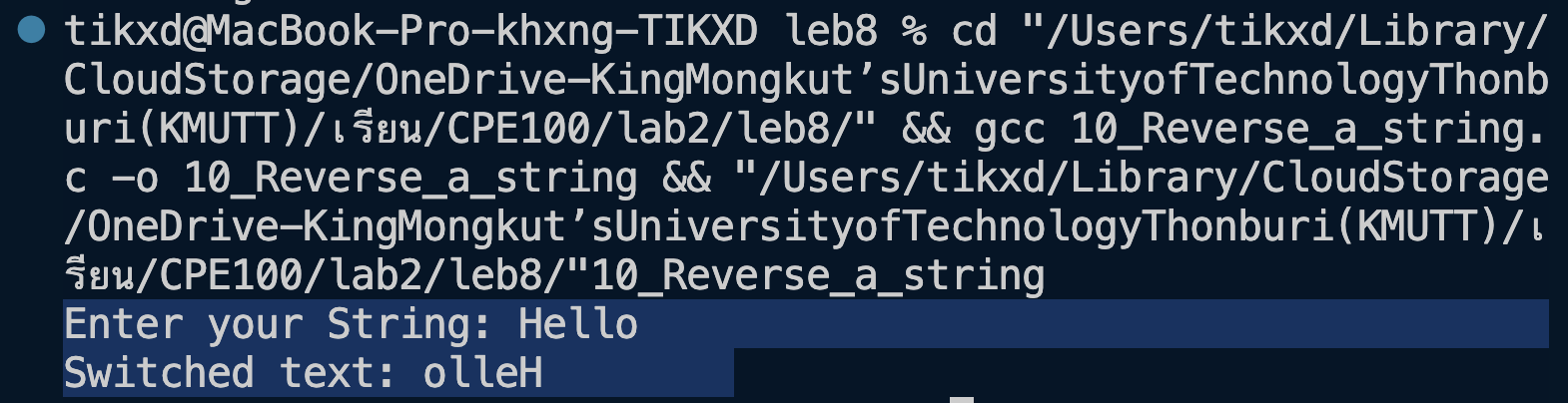
*reverseString*(*input*); *// JEDSADAPORN PANNOK NO.66070503410*

*printf*("\n");

*return* 0;

}

Output



Pascals Triangle

Pascal’s Triangle can be seen as the triangle of numbers where each number is the sum of the above two (except for the edges, which are all “1”)

*#include* <stdio.h>

*int* *calculatePascal*(*int* i, *int* j) {

*if* (j *==* 0 *||* j *==* i) {

*return* 1; *// The first and last element in each row is always 1*

} *else* {

*return* *calculatePascal*(i *-* 1, j *-* 1) *+* *calculatePascal*(i *-* 1, j);

}*// JEDSADAPORN PANNOK NO.66070503410*

}

*void* *printPascalsTriangle*(*int* n) {

*int* *i*, *j*;

*for* (*i* *=* 0; *i* *<* n; *i++*) {

*// Print leading spaces to center-align the triangle*

*for* (*j* *=* 0; *j* *<* (n *-* *i* *-* 1); *j++*) {

*printf*(" ");

}

*for* (*j* *=* 0; *j* *<=* *i*; *j++*) {

*printf*("%d ", *calculatePascal*(*i*, *j*));

}

*printf*("\n");

}

}

*int* *main*() {

*int* *n*;

*printf*("Enter your Number: ");

*scanf*("%d", *&n*);

*printPascalsTriangle*(*n*);

*return* 0;

}

Output

A screenshot of a computer

Description automatically generated

Palindrome Recursion

Recursive function to check if a string is palindrome. Given a string, write a recursive function that checks if the given string is a palindrome, else, not a palindrome.

1) If there is only one character in string, return true.

2) Else compare first and last characters and recur for remaining substring.

*#include* <stdio.h>

*#include* <stdbool.h>

*#include* <string.h>

*// Function to check if a string is a palindrome*

*bool* *Palindrome*(*const* *char* str*[]*) {

*int* *len* *=* *strlen*(str);

*for* (*int* *i* *=* 0; *i* *<* *len* */* 2; *i++*) {

*if* (str[*i*] *!=* str[*len* *-* *i* *-* 1]) {

*return* *false*;

}

}

*return* *true*;

}

*int* *main*() {*// JEDSADAPORN PANNOK NO.66070503410*

*char* *str*[100];

*printf*("Enter a string: ");

*scanf*("%s", *str*);

*if* (*Palindrome*(*str*)) {

*printf*("Yes\nReverse of %s is also %s.\n", *str*, *str*);

} *else* {

*printf*("No\nReverse of %s is not %s.\n", *str*, *str*);

}

*return* 0;

}

Output

A screen shot of a computer program

Description automatically generated

A screen shot of a computer screen

Description automatically generated