**PROBLEM SOLVING AND PROGRAMMING I - 2019**

**Individual Assignment Analysis And PseudoCode:**

**1.a)**

Read 3 positive integer numbers representing for the day, month and year. Then, the program will display the remain days of that month and the day of the year. Remember to check if that year is a leap year .

**Analysis :**

The program **reads** three integers day, month and year from the user and **outputs**

1. The remaining days of the given month
2. The day of the year

As the program needs to take in to consideration if the year is a leap year, the output calculation differs based on the input year

**Example 1: Leap year**

For the input 14-02-2020

The remaining days will be :15

The day of the year will be: 45

**Example 2: Non Leap Year**

For the input 14-02-2019

The remaining days will be : 14

The day of the year will be : 45 .

**PseudoCode:**

**START**

1) Read the value of the inputs day, month and year into **day** , **month** and year **respectively**

2) Check if the year is leap year

let **isLeapYear** = false

if **year** is exactly divided by 400 OR

**year** is exactly divided by 4 AND **year** is not exactly divided by 100

then isLeapYear = true;

3) Calculate remaining days and day of the year

if(**isLeapYear**)

Then **remainingDays** = 29 - day;

else

Then **remainingDays** = 28 - day;

let **dayOfTheYear** = 0

let i = 1

while i is less than month

if i is 2 and isLeapYear

**dayOfTheYear** = **dayOfTheYear** + 29;

else if i is 2 and not isLeapYear

**dayOfTheYear** = **dayOfTheYear** + 28;

else if i is 1 OR 3 OR 5 OR 7 OR 8 OR 10 OR 12

**dayOfTheYear** = **dayOfTheYear** + 31;

else

**dayOfTheYear** = **dayOfTheYear** + 30;

increment **i**

end while

print **remainingDays**

print **dayOfTheYear**

**END**

**1.b)**

Then, read 3 other integer numbers as another day, month and year. The program will find the difference in days between 2 date

**Analysis :**

The program **reads** anotherthree integers day, month and year from the user and **outputs**

1. The number of days between this date and the earlier date.

**Example 1:**

For the **input** of the dates 10-02-2019 and 21-03-2025

the output of the number of days between the dates are : 2,231

**PseudoCode:**

**START**

1) Read the date one input day, month and year into day1, month1 and year1 respecively

2) Read the date two input day, month and year into day2, month2 and year2 respecively

3) Calculate the number of days

let totalNumberOfDaysBetweenTheDates = 0

calculate dayOfTheYear1 using the algorithm given in 1a

if year1 is a leap year

then remainingDaysInYear1 = 366 - dayOfTheYear1;

else

then remainingDaysInYear1 = 365 - dayOfTheYear1;

totalNumberOfDaysBetweenTheDates = totalNumberOfDaysBetweenTheDates + remainingDaysInYear1;

calcudate dayOfTheYear2 using the algorithm given in 1a

totalNumberOfDaysBetweenTheDates = totalNumberOfDaysBetweenTheDates + dayOfTheYear2;

let i = year1 + 1

while i is less than or equal to year2 - 1

if i is a leap year

then totalNumberOfDaysBetweenTheDates = totalNumberOfDaysBetweenTheDates + 366

else

totalNumberOfDaysBetweenTheDates = totalNumberOfDaysBetweenTheDates+ 365

incrment i by 1

end while

print totalNumberOfDaysBetweenTheDates

**END**

2.

Write a program to mimic the ATM machine. First, the program asks users to enter the PIN number which is stored in the program. If the PIN is not correct, the program allows the user to retry 3 times. After 3 tries not successful, the program will stop. If the PIN correct, the program allows users to do following transactions. In the following, the bold type represents questions

the program types to the user. Each response to the transaction type question is a single character. Any attempted illegal transaction causes an error message. The program runs continuously until the user selects Exit option

**Analysis:**

1. The program takes PIN as the input from the user which is checked against a predefined PIN taken in the program
2. If pin is invalid then ask the user to enter the pin again, to max of 3 times.
3. The program then
   1. Either takes Transaction Type as O and Initial deposit of some amount.
   2. Or takes Transaction Type as B and prints the balance .
   3. Or takes Transaction Type as D and Deposit of some amount; and prints the old and new balance.
   4. Or takes Transaction Type as W and Withdrawal of some amount; and only prints new balance. The transaction type is allowed only if sufficient funds are available.
   5. Or takes Transaction Type and I and Computes interest at given % rate for given years and outputs the new balance.
   6. Or takes Transaction Type, E and Exit program.
4. The above step is done continuously until the user selects the E for Exiting the program

Example 1 : Assume the valid pin is 1234

Enter the pin: 1235

Enter the pin: 1234

Transaction Type?: O

Initial deposit?:150

Transaction Type?: B

Balance: 150

Transaction Type?: E

Exit.

Example 2 : Assume the valid pin is 1234

Enter the pin: 1234

Transaction Type?: O

Initial deposit?:150

Transaction Type?: W

Withdrawal amount : 50

New Balance: 100

Transaction Type?: E

Exit.

**PseudoCode:**

**START**

1) Let the pin be defined as PREDEFINED\_PIN = 1234

2) check the user entered pin

let numberOfRetries = 0

let userEnteredPin = 5678

while userEnteredPin is NOT valid and numberOfRetries <= 3 {

ask user to enter the pin

if userEnteredPin is equal to PREDEFINED\_PIN

then userEnteredPin is valid

increment numberOfRetries

}

3) Do transactions

if(userEnteredPin is valid) {

do {

Read the transactionType from the user

if transactionType == 'O' {

read the initial amount

} else if transactionType == 'B' {

print balance

} else if transactionType == 'D' {

read the deposit

print old-balance which is in balance

balance = balance + deposit

print new balance which is in balance

} else if transactionType == 'W' {

read the withdrawal amount

if balance >= withdrawal amount {

balance = balance - withdrawalAmount

}

print new-balance which is in balance

} else if transactionType == 'I' {

read the interest rate

read the number of years

balance = balance + ((interestRate \* balance)/ 100) numberOfYears

}

} while (transactionType not equal to 'E')

}

**END**

**3.)** In this exercise, your task is to develop a solution to convert a decimal number to an octal number and vice versa. Your program should allow users select which conversion they like.

**Analysis:**

1. Read the conversion type, octal to decimal or decimal to octal, from the user.
2. Based on the above input,
   1. If octal to decimal is selected, then ask the user to enter the octal number and convert into decimal number.
   2. If decimal to octal is selected, then ask the user to enter the decimal number and convert into octal number.

Example: 1 - Octal to decimal :

Octal Number : 24

Decimal Number : 20

Example: 2 decimal to Octal :

Decimal Number :36

Octal Number : 44.

**PSEUDO CODE**

**START**

1) Read the conversion type

2)

if conversion Type is decimal to octal

then {

Read the decimal number into decimalNumber

let octalNumber = 0;

let multiplicand = 1;

while(decimalNumber != 0) {

remainder = decimalNumber % 8

decimalNumber = decimalNumber / 8

octalNumber = remainder \* multiplicand + octalNumber;

multiplicand = multiplicand \* 10;

}

print octalNumber;

}

else { // octal to decimal

Read the octal number

let decimalNumber = 0

let multiplicand = 1;

while(octalNumber != 0) {

lastDigit = octalNumber % 10

octalNumber = octalNumber / 10

decimalNumber = decimalNumber + lastDigit \* multiplicand;

multiplicand = multiplicand \* 8;

}

print decimalNumber

}

**END**

**4.(a)** Develop a function that returns the sum of the proper divisors of a given integer. Proper divisors of an integer n are the positive divisors that are less than n. For instance, the proper divisors 8 are 1, 2, and 4, and hence the sum of them is 7. The proper divisors for 6 are 1, 2 and 3, and hence the sum of them is 6.

**Analysis:**

1. Take an integer number as input.
2. Find divisors of input.
3. Calculate the sum of all the divisors.
4. Return the sum.

Example: 1:

Input = 15

Divisors are 1,3,5.

Sum = 9.

Return sum.

**PSEUDOCODE**

**START**

1) read the number from the user

2)

let sumOfDevisors = 0;

for(int i = 1; i <= number / 2; i++) {

if number % i == 0

sumOfDevisors = sumOfDevisors + i;

}

print sumOfDevisors;

**END**

**4.(b)** Develop a function to test if a given positive integer is deficient, perfect or abundant. A positive integer is said to be a deficient, perfect or abundant number if the sum of its proper divisors is less than, equal to, or greater than the number. For instance, 8 is deficient because the sum of its proper divisors 1 + 2 + 4 < 8; 6 is perfect because 1 + 2 + 3 = 6; and 12 is abundant because 1 + 2 + 3 + 4 + 6 > 12 (NB: be aware that you need to make use of the function developed for exercise (a)).

**Analysis:**

1. Calculate the sum of divisors of the given number using the function define as part of 4a.
2. Compare the above sum with the given number.
   1. If given number is less than sum then the number is abundant.
   2. If given number is greater than sum then the number is deficient.
   3. If given number is equal to sum then the number is perfect.

Examples :

1. input = 8.

Output = deficient.

2. Input = 6.

Output = perfect.

3. Input = 12.

Output = abundant.

**PSEUDOCODE**

**START**

1) Read the number from the user

2) calculate the sumOfDevisors using the 4a procedure

3) if sumOfDevisors is less than number

then print the number is deficient

else sumOfDevisors is equal to the number

then print the number is perfect

else

the print number is abundant

**END**

**4.**(c) The xnor is a logic operation and a complement of an exclusive or (xor) operation. It can be applied to two binary strings of equal length where between two corresponding bits, 1 xnor 1 = 1, 1 xnor 0 = 0, 0 xnor 1 = 0 and 0 xnor 0 = 1. For instance, 1101011 xnor 1010101 = 1000001. Add a recursive function that takes two integer parameters simulating the two input binary numbers and returns the result of the xnor operation as an integer.

**Analysis:**

1. Read two binary strings of equal length.
2. Apply the xnor operator on the 2 input binary strings.

Examples :

1. For a input : 0101

0011

Output: 1001

2. For a input : 1101011

1010101

Output 1000001.

// using recursion

**PseudoCode:**

START

let result = 0,rem1,rem2;

calculateXnor(int number1, int number2) {

if number1 !=0 AND number2 != 0 {

rem1 = number1 % 10;

rem2 = number2 % 10;

if(rem1 == rem2) {

result = result \* 10 + 1;

} else {

result = result \* 10 + + 0;

}

calculateXnor(number1/10, number2/10);

}

else {

return result;

}

return result;

}

END

**4.(d)** Define a function that reverses the digits of a given positive integer. For instance, integer 123 becomes 321 after reversing. (NB: a recursive function receives more credit than an iterative function.)

**ANALYSIS:**

1. Given an input integer, return the reverse of the integer.

Examples :

Input : 123

Output : 321

Input : 456

Output : 654

**PSEUDOCODE**

**START**

1) Read the number to reverse

2)

let reveserNumber = 0;

while number is not 0

do {

remainder = number % 10

number = number / 10

reveserNumber = reveserNumber \* 10 + remainder

}

print reveserNumber

**END**