NAME: CHRYS SEAN T. SEVILLA **DATE:** OCTOBER 12, 2023

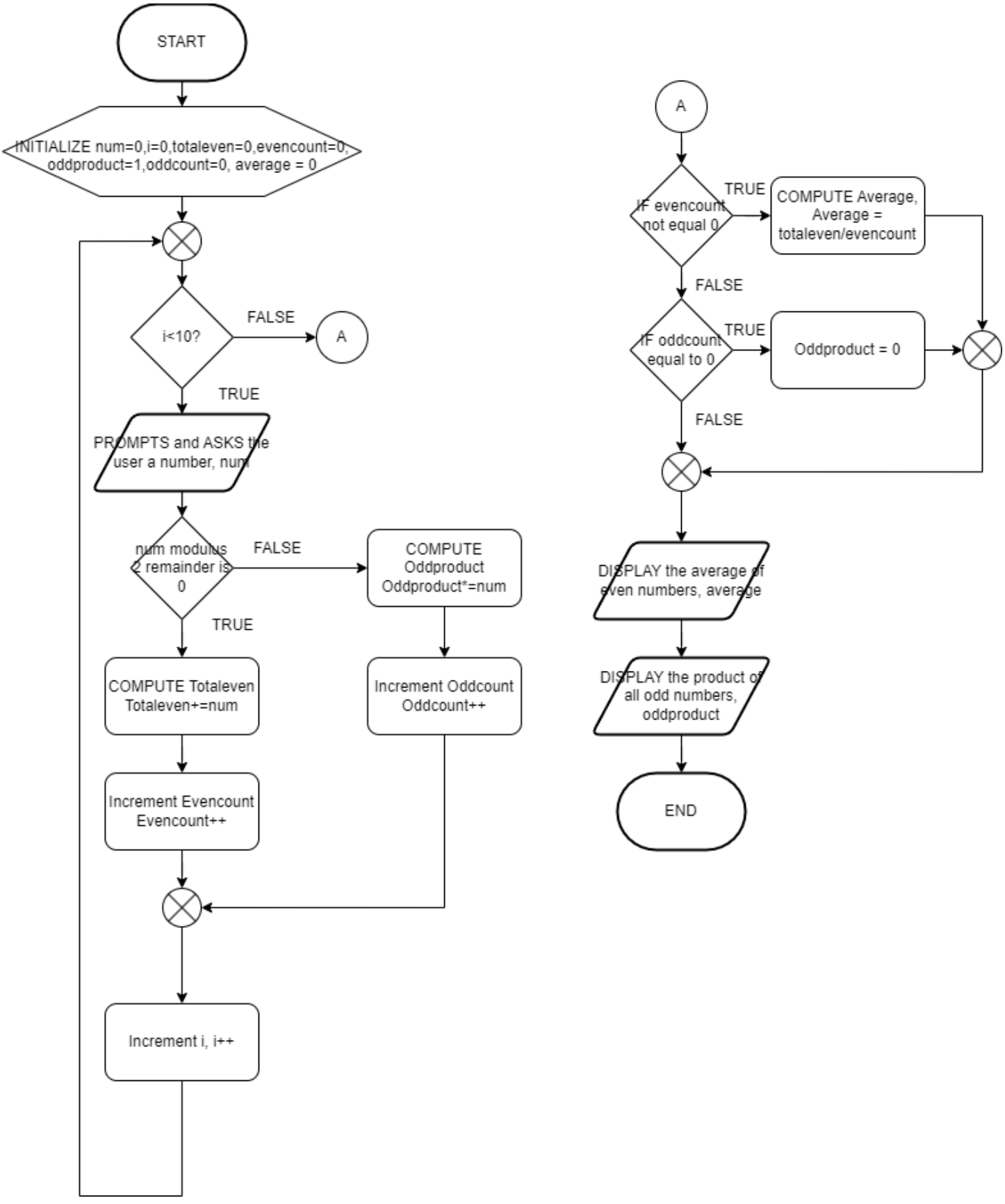
AlgoExercise 5.1

1. Average of Even Numbers and Product of Odd Numbers.
   Create a Program that will ask 10 integers and display the average of all even numbers and the product of all odd numbers entered by the user.

START

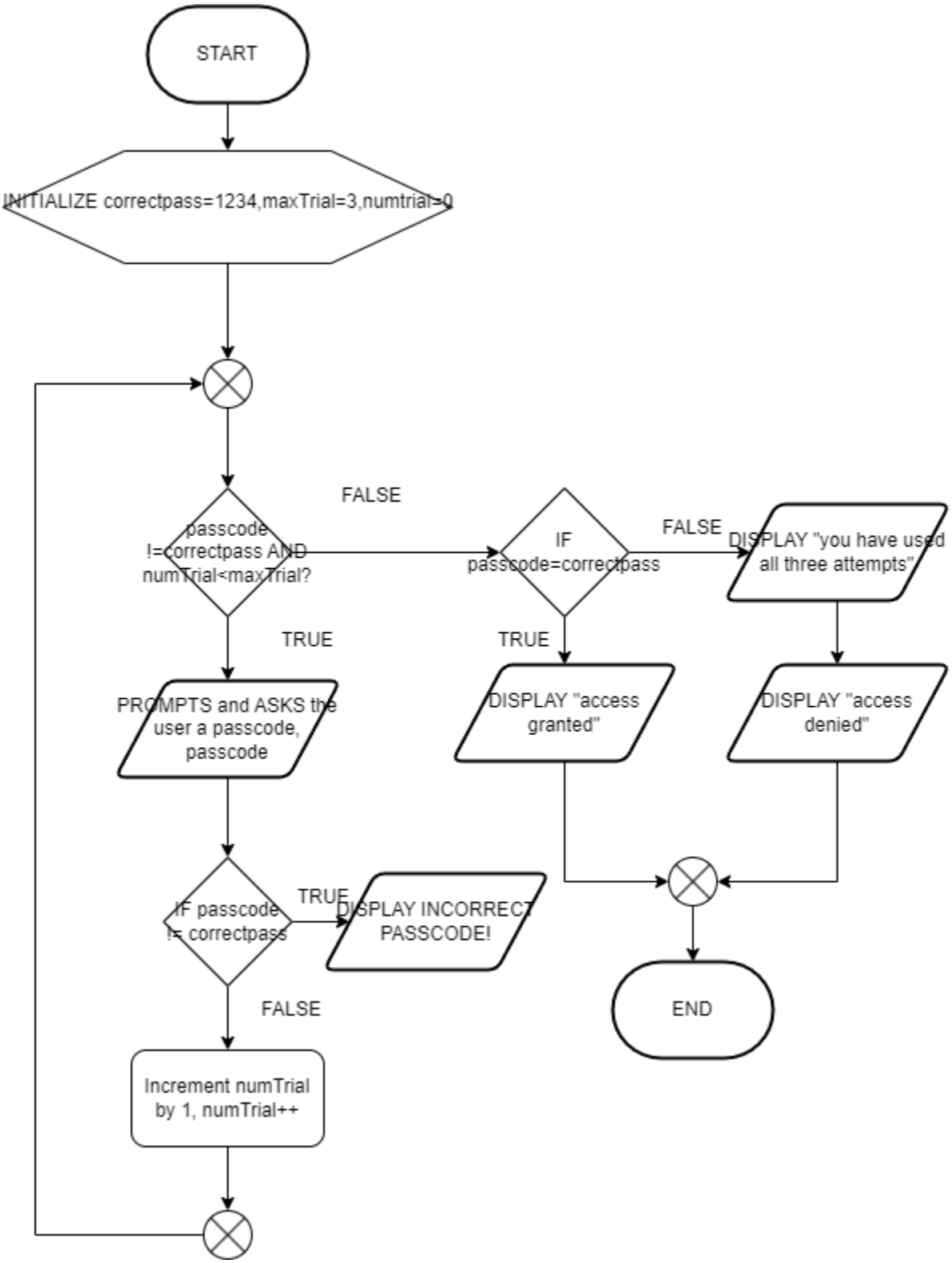
1. INITIALIZE num=0, i=0,totaleven=0,evencount=0,oddproduct=1,oddcount=0, average=0
2. FOR i=0 is less than 10 DO
   2.1. PROMPTS and ASKS the user number,num
   2.2. IF the number modulus 2 has no remainder THEN
      2.2.1. COMPUTE for Totaleven,Totaleven +=num
      2.2.2. INCREMENT Evencount, Evencount++
   2.3. ELSE num is odd
      2.3.1. COMPUTE Oddproduct, Oddproduct *= num
      2.3.2. INCREMENT oddcount,Oddcount++
   2.4. ENDIF
   2.5. Increment i,i++
3. ENDFOR
4. IF evencount not equal 0 THEN
   4.1. Average = totaleven/evencount
5. ENDIF
6. IF oddcount equal to 0 THEN
   6.1. Oddproduct = 0
7. ENDIF
8. DISPLAY the average of all even numbers in 2 decimal places, average
9. DISPLAY the product of all odd numbers, oddproduct

END


START
INITIALIZE num=0,i=0,totaleven=0,evencount=0, oddproduct=1,oddcount=0, average = 0
i<10?
FALSE
A
TRUE
PROMPTS and ASKS the user a number, num
num modulus 2 remainder is 0
FALSE
COMPUTE Oddproduct Oddproduct*=num
TRUE
COMPUTE Totaleven Totaleven+=num
Increment Oddcount Oddcount++
Increment Evencount Evencount++
Increment i, i++
A
IF evencount not equal 0
TRUE
COMPUTE Average, Average = totaleven/evencount
FALSE
IF oddcount equal to 0
TRUE
Oddproduct = 0
FALSE
DISPLAY the average of even numbers, average
DISPLAY the product of all odd numbers, oddproduct
END

2. Passcode Lock.
   Create a program that will ask the correct numeric passcode before the user can continue his task.
   START

1. INTIALIZE correctpass=1234, maxTrial = 3, numTrial=0
2. DO
   2.1. PROMPTS and ASKS Passcode, passcode
   2.2. Increment numTrial by 1, numTrial++
   2.3. IF passcode not equal to correctpass THEN
      2.3.1. DISPLAY incorrect passcode! Try Again
   2.4. ENDIF
3. WHILE passcode not equal correctpass and numTrial<maxTrial
4. IF passcode is equal to correctpass THEN
   4.1. DISPLAY access Granted
5. ELSE
   5.1. DISPLAY you have used all three attempts
   5.2. DISPLAY access Denied
6. ENDIF
   END

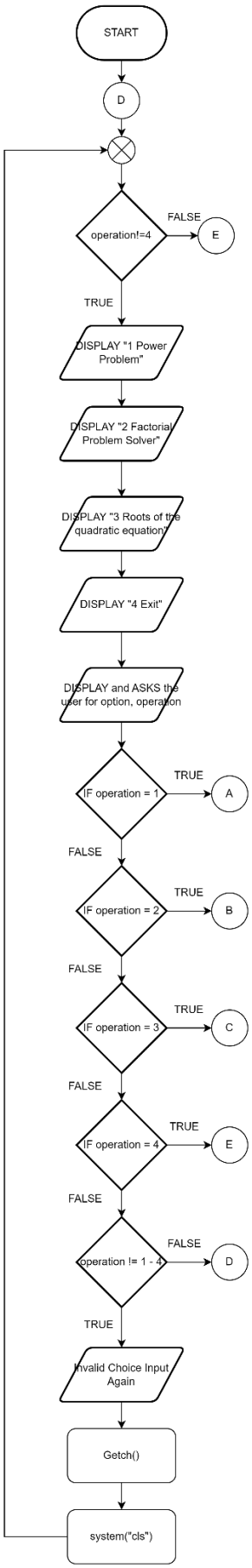

START
INITIALIZE correctpass=1234,maxTrial=3,numtrial=0
passcode !=correctpass AND numTrial<maxTrial?
FALSE
IF passcode=correctpass
FALSE
DISPLAY "you have used all three attempts"
TRUE
TRUE
PROMPTS and ASKS the user a passcode, passcode
DISPLAY "access granted"
DISPLAY "access denied"
IF passcode != correctpass
TRUE
DISPLAY INCORRECT PASSCODE!
FALSE
Increment numTrial by 1, numTrial++
END

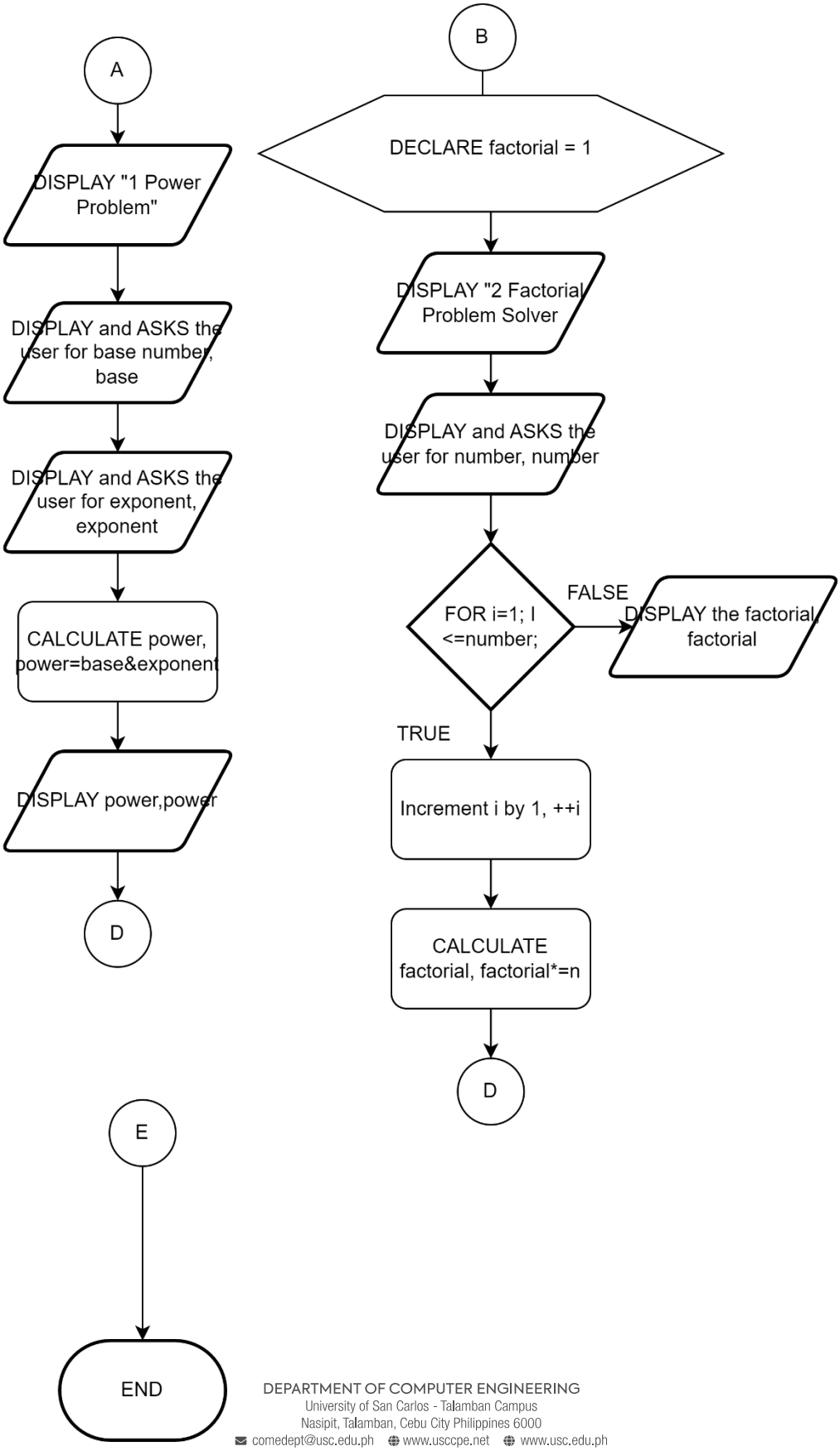
3. Problem Solver Menu. Create a program that lets the user choose an operation (power problem solver, factorial problem solver, or finding roots for quadratic equations using quadratic formula) from the menu.
   START

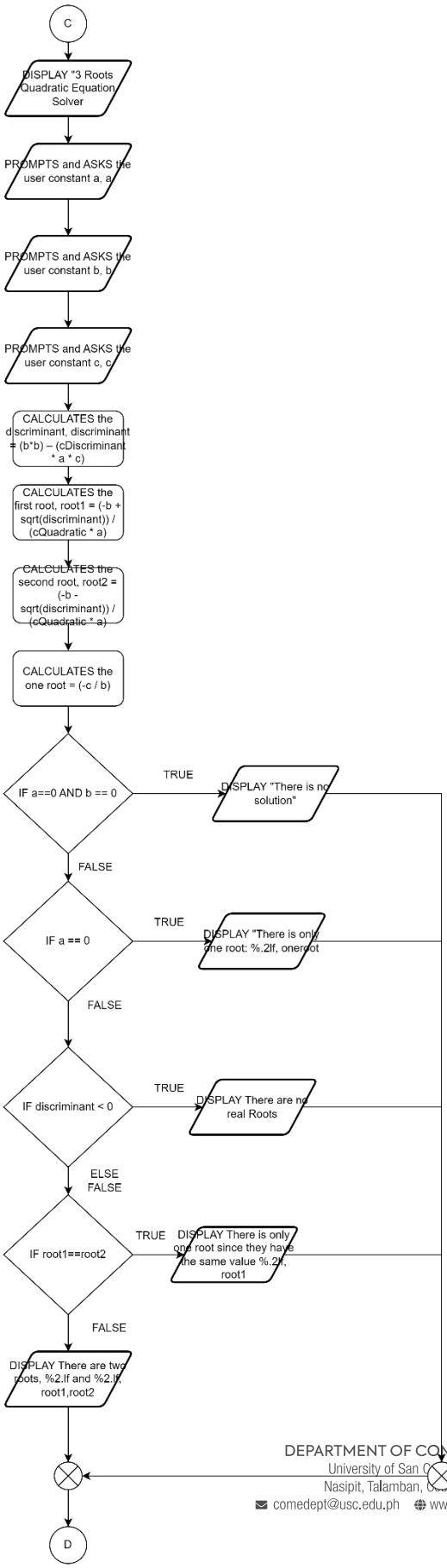
1. DO
   1.1. DISPLAY “1 Power Problem”
   1.2. DISPLAY “2 Factorial Problem Solver”
   1.3. DISPLAY “3 Roots of the quadratic equation solver”
   1.4. DISPLAY “4 Exit”
   1.5. PROMPT and asks the user for an option, operation
   1.6. IF operation = 1 THEN
      1.6.1. DISPLAY “Power Problem Solver”
      1.6.2. PROMPT and ASK base number, base
      1.6.3. PROMPT and ASK exponent, exponent
      1.6.4. CALCULATE power, power=base^exponent
      1.6.5. DISPLAY the power, power
   1.7. ENDIF
   1.8. IF operation = 2 THEN
      1.8.1. DECLARE factorial = 1
      1.8.2. DISPLAY “Factorial Problem Solver”
      1.8.3. PROMPT and ASK number, number
         1.8.3.1. FOR i=1; i is less than or equal to number
         1.8.3.2. INCREMENT i by one,++i
         1.8.3.3. CALCULATE factorial, factorial *= n;
      1.8.4. ENDFOR
      1.8.5. DISPLAY the factorial, factorial
   1.9. ENDIF
   1.10. IF operation = 3 THEN
      1.10.1. PROMPTS and ASKS the user constant a, a
      1.10.2. PROMPTS and ASKS the user constant b, b
      1.10.3. PROMPTS and ASKS the user constant c, d
      1.10.4. CALCULATES the discriminant, discriminant = (b*b) – (4 * a * c)
      1.10.5. CALCULATES the first root, root1 = (-b + sqrt(discriminant)) / (2 * a)
      1.10.6. CALCULATES the second root, root2 = (-b - sqrt(discriminant)) / (2 * a)
      1.10.7. CALCULATES the one root,oneroot = (-c / b)
      1.10.8. IF a==0 AND b ==0 THEN
         1.10.8.1. DISPLAY there is no solution
      1.10.9. ELSE IF a==0 THEN
         1.10.9.1. DISPLAY there is only one root: %.2lf, oneroot

1.10.10. ELSE IF discriminant is less than zero
1.10.10.1. DISPLAY there are no real roots
1.10.11. ELSE
1.10.11.1. IF root1==root2
1.10.11.1.1. DISPLAY There is only one root since they have the same value %.2lf, root1
1.10.11.2. ELSE
1.10.11.2.1. DISPLAY There are two roots, %.2lf and %2.lf, root1,root2
1.10.11.3. ENDIF
1.10.12. ENDIF
1.11. IF operation = 4
1.11.1. DISPLAY Exiting the program
1.12. ENDIF
1.13. IF operation != 1-4
1.13.1. DISPLAY Invalid Choice Try again
1.14. ENDIF
1.15. Getch()
1.16. System clear
2. While operation!=4

END


START
D
operation!=4
FALSE
E
TRUE
DISPLAY "1 Power Problem"
DISPLAY "2 Factorial Problem Solver"
DISPLAY "3 Roots of the quadratic equation"
DISPLAY "4 Exit"
DISPLAY and ASKS the user for option, operation
IF operation = 1
TRUE
A
FALSE
IF operation = 2
TRUE
B
FALSE
IF operation = 3
TRUE
C
FALSE
IF operation = 4
TRUE
E
FALSE
operation != 1 - 4
FALSE
D
TRUE
Invalid Choice Input Again
Getch()
system("cls")


A
DISPLAY "1 Power Problem"
DISPLAY and ASKS the user for base number, base
DISPLAY and ASKS the user for exponent, exponent
CALCULATE power, power=base&exponent
DISPLAY power,power
D
B
DECLARE factorial = 1
DISPLAY "2 Factorial Problem Solver
DISPLAY and ASKS the user for number, number
FOR i=1; I <=number;
FALSE
DISPLAY the factorial, factorial
TRUE
Increment i by 1, ++i
CALCULATE factorial, factorial*=n
D
E
END


C
DISPLAY "3 Roots Quadratic Equation Solver
PROMPTS and ASKS the user constant a, a
PROMPTS and ASKS the user constant b, b
PROMPTS and ASKS the user constant c, c
CALCULATES the discriminant, discriminant = (b*b) – (cDiscriminant * a * c)
CALCULATES the first root, root1 = (-b + sqrt(discriminant)) / (cQuadratic * a)
CALCULATES the second root, root2 = (-b - sqrt(discriminant)) / (cQuadratic * a)
CALCULATES the one root = (-c / b)
IF a==0 AND b == 0
TRUE
DISPLAY "There is no solution"
FALSE
IF a == 0
TRUE
DISPLAY "There is only one root: %.2lf, oneroot
FALSE
IF discriminant < 0
TRUE
DISPLAY There are no real Roots
ELSE
FALSE
IF root1==root2
TRUE
DISPLAY There is only one root since they have the same value %.2lf, root1
FALSE
DISPLAY There are two roots, %2.lf and %2.lf, root1,root2
D