**CSF213 Object Oriented Programming**

**Extra Practise Problems**

**Topics: static modifier, this keyword, Control Statements and Operators**

**Note: These problems can be done at students’ own pace outside the lab hours.**

**Exercise 1:**

Given the float variables a,b,c,root1,root2,and discriminant,write a code segment to determine whether the roots of a quadratic polynomial are real or complex (imaginary).If the roots are real, find them and assign them to root1 and root2. If they are complex, write the message “No real roots.”The formula for the solution to the quadratic equation is



The + means “plus or minus”and indicates that the equation has two solutions: one in which the result of the square root is added to **-** b and one in which the result is subtracted from **–** b .The roots are real if the discriminant (the quantity under the square root sign) is not negative.

**Exercise 2:**

People who deal with historical dates use a number called the Julian day to calculate the number of days between two events.The Julian day is the number of days that have elapsed since January 1,4713 B.C. For example,the Julian day for October 16,1956,is 2435763.

There are formulas for computing the Julian day from a given date, and vice versa. One very simple formula computes the day of the week from a given Julian day:

Day of the week = (Julian day + 1) % 7

where % is the Java modulus operator.This formula gives a result of 0 forSunday,1 for Monday, and so on, up to 6 for Saturday. For Julian day 2435763,the result is 2 (Tuesday).Your job is to write a Java application that requests and inputs a Julian day, computes the day of the week using the formula, and then displays the name of the day that corresponds to that number.

Expected output:

Enter a Julian day number and press Enter.

2451545

Julian day number 2451545 is a Saturday.

Enter a Julian day number and press Enter.

2451547

Julian day number 2451547 is a Monday.

**Exercise 3:**

You can compute the date for any Easter Sunday from 1982 to 2048 as follows (all variables are of type int):

a is year %19

b is year %4

c is year %7

d is (19\* a +24) %30

e is (2\* b +4\* c +6\* d +5) %7

Easter Sunday is March (22+ d + e)

For example,Easter Sunday in 1985 is April 7.

Write an application that inputs the year and outputs the date (month and day) of Easter Sunday for that year.

**Exercise 4:**

Write a program that takes three command line integers x, y, and z representing your two blackjack cards x and y, and the dealers face-up card z, and prints the "standard strategy" for a 6 card deck in Atlantic city. Assume that x, y, and z are integers between 1 and 10, representing an ace through a face card. Report whether the player should hit, stand, or split according to these strategy tables.

**Exercise 5:**

In 1769 Euler generalized Fermat's Last Theorem and conjectured that it is impossible to find three 4th powers whose sum is a 4th power, or four 5th powers whose sum is a 5th power, etc. The conjecture was disproved in 1966 by exhaustive computer search. Disprove the conjecture by finding positive integers a, b, c, d, and e such that a5 + b5 + c5 + d5= e5. Write a program that reads in a command line parameter N and exhaustively searches for all such solutions with a, b, c, d, and e less than or equal to N. No counterexamples are known for powers greater than 5, but you can join EulerNet, a distributed computing effort to find a counterexample for sixth powers.

**Exercise 6:**

The Universal Product Code (UPC) is a 12 digit code that uniquely specifies a product. The least significant digit d1(rightmost one) is a check digit which is the uniquely determined by making the following expression a multiple of 10:

(d1 + d3 + d5 + d7 + d9 + d11) + 3 (d2 + d4 + d6 + d8 + d10 + d12)

As an example, the check digit corresponding to 0-48500-00102 (Tropicana Pure Premium Orange Juice) is 8 since

(8 + 0 + 0 + 0 + 5 + 4) + 3 (2 + 1 + 0 + 0 + 8 + 0) = 50

and 50 is a multiple of 10. Write a program that reads in a 11 digit integer from a command line parameter, computes the check digit, and prints the full UPC. *Hint*: use a variable of type long to store the 11 digit number.

**Exercise 7:**

Write a program that finds the radii where the probability of finding the electron in the 4s excited state of hydrogen is zero. The probability is given by: *(1 - 3r/4 + r2/8 - r3/192)2 e-r/2*, where *r* is the radius in units of the Bohr radius (0.529173E-8 cm). Use Newton's method. By starting Newton's method at different values of *r*, you can discover all three roots. *Hint*: use initial values of r= 0, 5, and 13. *Challenge*: explain what happens if you use an initial value of r = 4 or 12.

**Exercise 8:**

In the game show *Let's Make a Deal*, a contestant is presented with three doors. Behind one door is a valuable prize, behind the other two are gag gifts. After the contestant chooses a door, the host opens up one of the other two doors (never revealing the prize, of course). The contestant is then given the opportunity to switch to the other unopened door. Should the contestant do so? Intuitively, it might seem that the contestant's initial choice door and the other unopened door are equally likely to contain the prize, so there would be no incentive to switch. Write a program to test this intuition by simulation. Your program should take an integer command-line argument *n*, play the game *n* times using each of the two strategies (switch or don't switch) and print the chance of success for each strategy. Or you can play the game here.

**Exercise 9:**

The factorial of a number n is n \* (n-1) \* (n-2) \* ... \* 2 \* 1. Stirling’s formula approximates the factorial for large values of n:

where  = 3.14159265 and e = 2.718282.

Write a Java application that inputs an integer value (but stores it into a double variable n), calculates the factorial of n using Stirling’s formula, assigns the (rounded) result to a long integer variable, and then displays the result appropriately labeled.

Depending on the value of n, you should obtain one of these results:

1. A numerical result.
2. If n equals 0, the factorial is defined to be 1.
3. If n is less than 0, the factorial is undefined.
4. If n is too large, the result exceeds Long.MAX\_VALUE.