**Program Projects 1 CSC 123 Spring 2021**

Notes: All homework must be submitted via e-mail. All parts of assignment must be submitted in a single e-mail with multiple attachments when required.

E-mail address is:

[csc123csudh@gmail.com](mailto:csc123csudh@gmail.com)

Each program or class is to be submitted in a separate file with the file name being the class name with extension .java as shown below. I only need the source file.

**GeneralAverage.java**

All solution will be posted on the web site after the due date.

1. **LastDigit (10)**

Write a program that prompts a user for an integer *n* > 0 and determines the last digit of 3*n*. *Hint:* The last digit depends on the value *n*%4. If *x* = *n*%4, then the last digit of 3*n* is:

-2*x*3 + 8*x*2 - 4*x* + 1.

Sample output:

Please the power of three: 1

The last digit of 3 to the power 1 is 3

Please the power of three: 2

The last digit of 3 to the power 2 is 9

Please the power of three: 3

The last digit of 3 to the power 3 is 7

Please the power of three: 4

The last digit of 3 to the power 4 is 1

1. **CalculatePi (10)**

Calculate the value of π from the infinite series:

π = 4 -4/3 + 4/5 -4/7 + 4/9 -4/11 + ….

Calculate by using the first 200000 terms and printing the result for each 2000Th number in a table. Print correct to 8 decimal places

The value of pi after summing 2000 terms is 3.14109265

The value of pi after summing 4000 terms is 3.14134265

The value of pi after summing 6000 terms is 3.14142599

The value of pi after summing 8000 terms is 3.14146765

The value of pi after summing 10000 terms is 3.14149265

The value of pi after summing 12000 terms is 3.14150932

The value of pi after summing 14000 terms is 3.14152123

The value of pi after summing 16000 terms is 3.14153015

The value of pi after summing 18000 terms is 3.14153710

The value of pi after summing 20000 terms is 3.14154265

The value of pi after summing 22000 terms is 3.14154720

The value of pi after summing 24000 terms is 3.14155099

The value of pi after summing 26000 terms is 3.14155419

The value of pi after summing 28000 terms is 3.14155694

The value of pi after summing 30000 terms is 3.14155932

The value of pi after summing 32000 terms is 3.14156140

The value of pi after summing 34000 terms is 3.14156324

The value of pi after summing 36000 terms is 3.14156488

The value of pi after summing 38000 terms is 3.14156634

The value of pi after summing 40000 terms is 3.14156765

The value of pi after summing 42000 terms is 3.14156884

The value of pi after summing 44000 terms is 3.14156993

The value of pi after summing 46000 terms is 3.14157091

The value of pi after summing 48000 terms is 3.14157182

The value of pi after summing 50000 terms is 3.14157265

The value of pi after summing 52000 terms is 3.14157342

The value of pi after summing 54000 terms is 3.14157414

The value of pi after summing 56000 terms is 3.14157480

The value of pi after summing 58000 terms is 3.14157541

The value of pi after summing 60000 terms is 3.14157599

The value of pi after summing 62000 terms is 3.14157652

The value of pi after summing 64000 terms is 3.14157703

The value of pi after summing 66000 terms is 3.14157750

The value of pi after summing 68000 terms is 3.14157795

The value of pi after summing 70000 terms is 3.14157837

The value of pi after summing 72000 terms is 3.14157876

The value of pi after summing 74000 terms is 3.14157914

The value of pi after summing 76000 terms is 3.14157950

The value of pi after summing 78000 terms is 3.14157983

The value of pi after summing 80000 terms is 3.14158015

The value of pi after summing 82000 terms is 3.14158046

The value of pi after summing 84000 terms is 3.14158075

The value of pi after summing 86000 terms is 3.14158103

The value of pi after summing 88000 terms is 3.14158129

The value of pi after summing 90000 terms is 3.14158154

The value of pi after summing 92000 terms is 3.14158178

The value of pi after summing 94000 terms is 3.14158202

The value of pi after summing 96000 terms is 3.14158224

The value of pi after summing 98000 terms is 3.14158245

The value of pi after summing 100000 terms is 3.14158265

The value of pi after summing 102000 terms is 3.14158285

The value of pi after summing 104000 terms is 3.14158304

The value of pi after summing 106000 terms is 3.14158322

The value of pi after summing 108000 terms is 3.14158339

The value of pi after summing 110000 terms is 3.14158356

The value of pi after summing 112000 terms is 3.14158373

The value of pi after summing 114000 terms is 3.14158388

The value of pi after summing 116000 terms is 3.14158403

The value of pi after summing 118000 terms is 3.14158418

The value of pi after summing 120000 terms is 3.14158432

The value of pi after summing 122000 terms is 3.14158446

The value of pi after summing 124000 terms is 3.14158459

The value of pi after summing 126000 terms is 3.14158472

The value of pi after summing 128000 terms is 3.14158484

The value of pi after summing 130000 terms is 3.14158496

The value of pi after summing 132000 terms is 3.14158508

The value of pi after summing 134000 terms is 3.14158519

The value of pi after summing 136000 terms is 3.14158530

The value of pi after summing 138000 terms is 3.14158541

The value of pi after summing 140000 terms is 3.14158551

The value of pi after summing 142000 terms is 3.14158561

The value of pi after summing 144000 terms is 3.14158571

The value of pi after summing 146000 terms is 3.14158580

The value of pi after summing 148000 terms is 3.14158590

The value of pi after summing 150000 terms is 3.14158599

The value of pi after summing 152000 terms is 3.14158607

The value of pi after summing 154000 terms is 3.14158616

The value of pi after summing 156000 terms is 3.14158624

The value of pi after summing 158000 terms is 3.14158632

The value of pi after summing 160000 terms is 3.14158640

The value of pi after summing 162000 terms is 3.14158648

The value of pi after summing 164000 terms is 3.14158656

The value of pi after summing 166000 terms is 3.14158663

The value of pi after summing 168000 terms is 3.14158670

The value of pi after summing 170000 terms is 3.14158677

The value of pi after summing 172000 terms is 3.14158684

The value of pi after summing 174000 terms is 3.14158691

The value of pi after summing 176000 terms is 3.14158697

The value of pi after summing 178000 terms is 3.14158704

The value of pi after summing 180000 terms is 3.14158710

The value of pi after summing 182000 terms is 3.14158716

The value of pi after summing 184000 terms is 3.14158722

The value of pi after summing 186000 terms is 3.14158728

The value of pi after summing 188000 terms is 3.14158733

The value of pi after summing 190000 terms is 3.14158739

The value of pi after summing 192000 terms is 3.14158745

The value of pi after summing 194000 terms is 3.14158750

The value of pi after summing 196000 terms is 3.14158755

The value of pi after summing 198000 terms is 3.14158760

The value of pi after summing 200000 terms is 3.14158765

1. **HarmonicMean (25)**

If it takes one hose 12 hours to fill a pool, and another hose 4 hours, then together they fill the pool in (2 \* 4 \* 12) / (4 +12) = 6 hours. The *harmonic mean* of two positive numbers *a* and *b* is 2*ab*/(*a* + *b*).

Write a method **double harmonicMean(int x, int y)** (5)

that returns the harmonic mean of *a* > 0 and *b* > 0.

Write another method **double** **arithmeticMean(int x, int y)**, (5)

that returns the average of *a* and *b*.

Finally, include a third method **double geometricMean(int x, int y)** (5)

that returns the geometric mean of *a* and *b*, that is, the square root of *a* \* *b*.

Test your methods in a **main** program (10) that reads two positive integers and displays their harmonic mean, arithmetic mean, and geometric mean. For example, if *a* and *b* have values 12 and 4, the harmonic mean is 6.0, the arithmetic mean is 8.0, and the geometric mean is √48 = 6.928. Did you notice that the harmonic mean times the arithmetic mean equals the square of the geometric mean? This identity might be helpful to you when you design your methods.

Please enter two integers greater than zero: 4 12

Harmonic Mean is: 6.0000000

Arithmetic Mean is: 8.0000000

Geometric Mean is: 6.9282032

Please enter two integers greater than zero: 3 10

Harmonic Mean is: 4.6153846

Arithmetic Mean is: 6.5000000

Geometric Mean is: 5.4772256

1. **ArrayOperations2D (55)**

Write a main program (5) that reads in the dimensions of a two-dimensional array and creates a two-dimensional array with those rows and columns, and then call the methods below in the order listed to produce the output described below.

The additional method are:

* **fillRandom**. (5) Accepts a reference to a two-dimensional int array and fills it with random integers from 0 to 99. Has a void return. (you can use Math.random() or a Random class object here.
* **formatPrint**. (5) This method should accept a reference to two-dimensional array and print it out row by row. Has a void return.
* **getTotal**. (5) This method should accept a reference to two-dimensional array as its argument and return the total of all the values in the array as an int.
* **getAverage**. (5) This method should accept a two-dimensional array as its argument and return the average of all the values in the array as a double. It calls calls getTotal and getElementCount in performing this task.
* **getRowTotal**. (5) This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The method should return the total of the values in the specified row as an int.
* **getColumnTotal**. (5) This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a column in the array. The method should return the total of the values in the specified column. as an int
* **getHighestInRow**. (5) This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the subscript of a row in the array. The method should return the highest value in the specified row of the array as an int.
* **getLowestInRow**. (5) This method should accept a two-dimensional array as its first argument and an integer as its second argument. The second argument should be the sub- script of a row in the array. The method should return the lowest value in the specified row of the array.
* **getElementCount**. (5) This method should accept a two-dimensional array and returns the total number of elements in the array as an int.

Demonstrate each of the methods in this program. Each (except for getElementCount, are called from main.

The main program will request the number of rows and columns as input, creates the two-dimensional array, and first calls fillRandom. A sample output is:

Please enter the number of rows and columns in a two dimensional array: 4 5

78 65 72 30 95

60 71 88 41 73

32 74 47 70 27

59 91 80 81 87

Processing the int array.

Total : 1321

Average : 66.05

Total of row 0 : 340

Highest in row 0 : 95

Lowest in row 0 : 30

Total of row 1 : 333

Highest in row 1 : 88

Lowest in row 1 : 41

Total of row 2 : 250

Highest in row 2 : 74

Lowest in row 2 : 27

Total of row 3 : 398

Highest in row 3 : 91

Lowest in row 3 : 59

1. **SlotMachine (25)**

In main: (15)

Ask for the number of times to run the game.

Read in a seed as type long.

For each time you run the game

Increment seed by 1

Call runGame with the incremented seed.

Based on the returned value keep count of the wins and losses.

Print out the number of wins as an integer and the percentage of wins to decimal places as below:

The number of games won is *numWins* and the percentage of wins *is pctWins*%.

In runGame: 910)

Use the Random class to do the following

Create a Random object.

Set the seed based on the input value.

Generate 3 integers between 1 and 5.

Return true if all three match (a win) or false otherwise.

Sample output:

Please enter the number of times to run the game: 5

Please enter the starting seed as an integer: 2

The number of games won is 1 and the percentage of wins is 20.00%.

Please enter the number of times to run the game: 100

Please enter the starting seed as an integer: 2

The number of games won is 3 and the percentage of wins is 3.00%.

Please enter the number of times to run the game: 1000

Please enter the starting seed as an integer: 2

The number of games won is 47 and the percentage of wins is 4.70%.

1. **String Rotation (20)**

Write a program that rotates a given string n characters to the right (program can be done all in main. . For example, if the input to your program is

rotatemeplease 4, then the output is

easerotatemepl

Sample I/O

Please enter a String for rotation

InterestinJava

Please enter the number of characters to rotate. Rotation must be greater than or equal to 0

-2

Please enter the number of characters to rotate. Rotation must be greater than or equal to 0

1

When the original String "InterestinJava" is rotated 1 the rotated String is "aInterestinJav"

Please enter a String for rotation

InterestinJava

Please enter the number of characters to rotate. Rotation must be greater than or equal to 0

13

When the original String "InterestinJava" is rotated 13 the rotated String is "nterestinJavaI"

Please enter a String for rotation

InterestinJava

Please enter the number of characters to rotate. Rotation must be greater than or equal to 0

27

When the original String "InterestinJava" is rotated 13 the rotated String is "nterestinJavaI"

1. **RemoveBlanksFromStrings** (**50 Points)**

This program will read in lines one at a time from a file and write output to a file as described below. You can use the file withblanks.txt to test your program, or you can create your own input file. The withblanks.txt file is found under the Project Solutions/Project 1 tab.

The method **main** does the following: (10)

1. Creates a Scanner to read from the keyboard.
2. Reads in the name of the physical file to be read from (Hint: use next, not nextLine).
3. Calls **createFileScanner** with the name of the physical file to read from, receiving the reference to the Scanner as the return value in a Scanner reference variable.
4. Reads in the name of the physical file to write to.
5. Calls **createPrintWriter** with the name of the physical file to write to, receiving the reference to the PrintWriter as the return value in a PrintWriter reference variable.
6. Create a one-dimensional String array of length 100.
7. Call **readInLines** with a reference to the Scanner created to read from the file, and a reference to the String array, returning an int with the number of lines read in.
8. Call **removeBlanks** with a reference to the String array described above and to an int with the number of lines read into the array as described above, and then perform the functions described below, accepting a reference to the String array as defined below.
9. Write the lines in the returned String array to the output file created in this program.
10. Close the Scanner and PrintWriter objects.

**createFileScanner**: (10)

1. Accepts as a String the name of the physical file.
2. Creates a Scanner throwing an exception if the physical file doesn’t exist.
3. Returns a reference to the Scanner.

**createPrintWriter**: (10)

1. Accepts as a String the name of the physical file.
2. Creates a PrintWriter.
3. Returns a reference to the PrintWriter.

**readInLines**: (10)

1. Accepts a reference to the Scanner for the physical file, and to a String Array.
2. Read in the up to the first 100 lines in the file and place a reference to each line in the String array.
3. Return an int with the number of lines read in.

**removeBlanks**: (10)

1. Accepts two parameters including a reference to the String array described above and to an int with the number of lines read into the array.
2. Create a second String array with the length equal to the number of lines read into the first array.
3. Removes the blanks from each line and places a reference to the line in the corresponding cell of the second String array.
4. Return a reference to the second String array

**A Sample of Input and Output**

**Input file:**

To be or not to be.

HowAre You?

Bravaco Gaddis Mr. Rogers

1 2 3 4 Let us Cheer a Little More.

Taking tests is lots of fun

Better than sitting in the sun.

Today tomorrow and beyond

Have yourself a programming marathon

! @ # $ % % ^ & \* ( ) \_

**Output File:**

Tobeornottobe.

HowAreYou?

BravacoGaddisMr.Rogers

1234LetusCheeraLittleMore.

Takingtestsislotsoffun

Betterthansittinginthesun.

Todaytomorrowandbeyond

Haveyourselfaprogrammingmarathon

!@#$%%^&\*()\_

1. **CaeasarCipherFromFile (30)**

Rewrite the CaesarCipher program to accept input from a file (i.e. caesar\_input.txt found in Project Solutions/Project 1 tab) and write the output to a file.

Read in the shift for coding only once from the terminal.

Use nextLine() to read in a line at a time.

You can use caesar\_input.txt and see the result with a shift of 3 in caesar\_output.txt. Both files are posted in Project Solutions/Project 1 tab.

1. **NameSearch (50)**

In the following problem use these two files which have been placed in Project Solutions/Project 1 tab.

• GirlNames.txt – This file contains a list of the 200 most popular names given to girls born in the United States for the years 2000 through 2009.

• BoyNames.txt – This file contains a list of the 200 most popular names given to boys born in the United States for the years 2000 through 2009.

In **main**: (20)

Get the names of the two files with the lists. Call **getArray** with the name of the file (a String). Return a String array with the names in the file.

Ask the user if they wish to continue querying against the list.

The user should be able to enter a boy’s name, a girl’s name, or both, and the application will display messages indicating whether the names were among the most popular. To determine if the name is on the list call isFound with a reference to the name being searched for and the array of names as the two parameters. Ignore capitalization in your compares and in any other inputs.

In **getArray**: (10)

Create a File object and a Scanner for that file object.

Call **getNumNames** to determine the number of items in the file. Send **getNumNames** the reference to the File object.

Read the list of names from the file into an array and then return that array.

Close the Scanner

In **getNumNames**: (10)

Accept the File object name and count the number of items in the file.

Return the number of items in the file.

In **isFound**: (10)

Accept the name being searched for and the array with all the names. Return true if the name is found, otherwise false. Ignore capitalization in your compares.

Below you see a sample run that test. Be sure to search on first and last name in list, as well as on same name in the middle. Also check for names that aren’t on the list.

Please enter the name of the boy's file: BoyNames.txt

Please enter the name of the girl's's file: GirlNames.txt

Do you want to search for names? Enter Y or N: Y

Enter a boy's name, or N if you do not wish to enter a boy's name: Jacob

Enter a girl's name, or N if you do not wish to enter a girl's name: Emily

Jacob is one of the most popular boy's names.

Emily is one of the most popular girl's names.

Do you want to search for names? Enter Y or N: Y

Enter a boy's name, or N if you do not wish to enter a boy's name: Martin

Enter a girl's name, or N if you do not wish to enter a girl's name: Jasmin

Martin is one of the most popular boy's names.

Jasmin is one of the most popular girl's names.

Do you want to search for names? Enter Y or N: Y

Enter a boy's name, or N if you do not wish to enter a boy's name: Devon

Enter a girl's name, or N if you do not wish to enter a girl's name: Camila

Devon is one of the most popular boy's names.

Camila is one of the most popular girl's names.

Do you want to search for names? Enter Y or N: Y

Enter a boy's name, or N if you do not wish to enter a boy's name: Hubert

Enter a girl's name, or N if you do not wish to enter a girl's name: Delaney

Hubert is not one of the most popular boy's names.

Delaney is one of the most popular girl's names.

Do you want to search for names? Enter Y or N: Y

Enter a boy's name, or N if you do not wish to enter a boy's name: Edgar

Enter a girl's name, or N if you do not wish to enter a girl's name: Helena

Edgar is one of the most popular boy's names.

Helena is not one of the most popular girl's names.

Do you want to search for names? Enter Y or N: Y

Enter a boy's name, or N if you do not wish to enter a boy's name: Henrique

Enter a girl's name, or N if you do not wish to enter a girl's name: Isabella

Henrique is not one of the most popular boy's names.

Isabella is one of the most popular girl's names.

Do you want to search for names? Enter Y or N: N