**1– Zoo Tickets**

Pete at Pete’s Petting Zoo wants a program to work out what to charge groups

of visitors to his children’s zoo. An adult’s ticket costs $10, a child’s ticket

cost $5 and a family ticket (for 2 adults and 2 children) costs $26.

Write a program that asks the user to enter the number of adults and the

number of children in the group and displays to screen the cheapest selection

of tickets and the total cost. A sample execution of the program is shown here.

**Number of adults: 1**

**Number of children: 2**

**Number of family tickets: 0**

**Number of adult tickets: 1**

**Number of child tickets: 2**

**Total cost: $20**

\* \* \*

**2– Counting Occurrences of Search Word**

Note: For this question, a “word” is simply taken as a sequence of characters,

which may occur as a “stand-alone word” or part of another word.

**a.** Write a program that asks for a string and a search word. The program

then determines and displays how many time the search word appears in

the string.

**b.** The following program provides a function which takes a file name, reads

the file and returns a string containing the content of the file.

Complete the program to ask the user for a file name and a search word,

and then determine and display how many times the search word appears

in the file.

**# This function read a file and return a string containing**

**# the text of the file.**

**def readFile(fileName):**

**inFile = open(fileName)**

**text = inFile.read();**

**return text**

**# An example showing how to call the function provided.**

**# Comment out lines 10 and 11 once you know how they work.**

**lines = readFile("test.txt")**

**print(lines)**

**# ADD a statement to ask the user for a file name of your choice**

**# ADD a statement to ask for a search word**

**# ADD statements to determine and display how many times the search word**

**# appears in the file. Make the search case-insensitive.**

(Question you may ask and my answer:)

**Q:** This Question 2 has two parts. Should I submit one program or two

programs?

**A:** Please submit the answers for both parts in *one file* (though the code in

the file may be a little messy).

**3 – String Representing Floating-Point Value**

Write a program that prompts the user for a floating-point value to be entered

as decimal number (with a decimal point) or an integer (without a decimal

point).

The program then determines if the string entered by the user actually repre-

sents a floating-point value or not.

**Some test cases:**

Valid cases:

• **"123.45"**

• **"-123.45"**

• **"+123.45"**

• **".123"**

• **"123."**

• **"123"**

• **"+123"**

• **"-123"**

Invalid cases:

• **"123.45."**

• **"12a3"**

• **"."**

**4 – Millionaire Plan**

Bob has a plan to be a millionaire. In month 1, he saves 1 dollar. In month 2,

he saves 2 dollars. In month 3, he saves 4 dollars, and so on. In general, for

each month, he saves twice the amount of the previous month.

Write a program to answer these two questions:

**a.** How much money will Bob save after 1 year (12 months)?

**b.** When (after how many months) he will become a millionaire?

**Q:** Question 4 asks me to write a program to answer two questions (which is a

slight modification of the lab question). How should I arrange the two parts?

**A:** Please put both answers in the same file. Please also make sure that the

answers are independent of each other. That is, *each can be run separately by*

*itself* when the other answer is commented out.

**5 – Rounding to Nearest Five Cents**

**a.** Write a function, named **adjust**, that takes a number of cents between 0

and 9, inclusive, rounds it off to the nearest 5 cents by the following rules,

and returns the result:

• 1 and 2 cents are rounded off to 0

• 3, 4, 6 and 7 cents are rounded off to 5

• 8 and 9 cents are rounded off to 10 cents

**b.** Write a function, named **roundOff**, that takes an amount of money in

dollars, and rounds it off to the nearest 5 cents and returns the result. The

amount is entered as a decimal number with 0, 1 or 2 decimal places. The

rounding off is based on the rules given above.

**Q:** Must I name the functions exactly as asked?

**A:** Yes, you must. This conformity will be important when I test your program.

**Q:** This question does not require me to do the testing. Should I include it?

**A:** Though not required, it would be nice to include some tests showing how

you would test the functions.

**6 – Square Root**

An algorithm to calculate the square root of a number, popularly known as

the Babylonian algorithm, can be described as follows.

Suppose x is the number we want to calculate the square root for.

• Step 1: First, we make an initial guess, for example, we can take

guess = 1

2x

• Step 2: From the existing guess value, a better guess, here denoted by guess

′

(pronounced “guess prime”), can be calculated as

guess

′ = 1

2 (guess + x

guess

)

• Step 3: Repeat Step 2 until the square of the value of guess is close enough

to x . We can take that two values are practically equal if their difference is

less than 10−12.

Write a function, called **squareRoot**, that takes a float value (which you can

assume to be positive), calculates its square root, and returns it.

Notes: (1) Your function is not required to check that the argument is positive.

You can assume this; (2) Though not required, you should include some tests

to indicate how you preform the testing.

**7 – Credit Card Checking**

(From Cay Horstmann’s Big Java)

A credit card number is a *string* of 16 digits. The last digit is known as the

check digit. It can be used to check the validity of a credit card number,

against transcription error for example, as described below. For simplicity, we

will describe the procedure using 8-digit number instead of the actual 16-digit

one. We will also take the number 43589795 as an example.

• Starting from the rightmost digit, form the sum of of every other digit.

For the example, the sum would be 5 + 7 + 8 + 3 = 23.

• Double each of the digits that are not included in the preceding step.

Add all the digits of the resulting numbers. For the example, we would

take the digits 9, 9, 5, 4, and double them up to get 18, 18, 10, 8, and

then add up the digits 1 + 8 + 1 + 8 + 1 + 0 + 8 = 27

• Add up the sums of the two preceding steps. If the last digit of the result

is 0, the number is valid. For the example, we get 23 + 27 = 50, and

the number is valid.

Write a program to implement this algorithm. Ask the user for an 8-digit

number. Determine, and display, if the number is valid or not.

**Q:** The question does not ask me to write any function, but if defining a

function can make my calculation easier to express. Should I define such a

function and use it?

**A:** Yes! Please go for it.

**Q:** In the above example, the final *check sum* is 50, and therefore we conclude

that the credit card number is valid. What if the last digit of the check sum

is not 0. Can we determine what the check digit should be to make the credit

card number valid?

**A:** Yes, we can. Let **CD** denote the check digit and **E** denote the last digit of

the check sum. The check digit we want can be calculated as

**new CD = (old CD - E + 10) %10**

Try it out if you wish.

6

**8– Date and Calendar**

**a.** Define a function called **isLeapYear** that takes a year as an integer and

returns **True** if it is a leap year and **False** otherwise.

A year, between 1800 and 20000, is a leap year if

• It is divisible by 4 but not by 100 *or*

• It is divisible by 400

For this function, you can assume that the year is between 1800 and 20000.

You do not need to check it.

**b.** Define a function called **daysInMonth** that takes *a month of a year* and

returns the *number of days* for that month. The function takes an integer

for the month (1 for January, 2 for February, etc.) and a year (an integer

between 1800 and 20000).

For this function, you can assume that the month is between 1 and 12 and

the year is between 1800 and 20000. You do not need to check them.

**c.** Define a function called **isValidDate** that takes three integers for a day,

a month and a year (in that order) and returns **True** if the thee integers

represent a valid date and **False** otherwise.

*Among other things*, this function must check that the day is between 1

and 31, the month is between 1 and 12, and the year is between 1800 and

20000. The function must return **False** if any of these conditions is not

satisfied.

**d.** Let day, month and year be the day, the month and the year of a *date*.

The *day of the week* this date falls in can be determined by the following

method of calculation (given by the famous mathematician Carl Friedrich

Gauss):

**x = year - (14 - month)/12**

**y = x + x/4 - x/100 + x/400**

**z = month + 12 \* ((14 - month) /12) -2**

**dow = (day + y + (31 \* z)/12) % 7**

where the division is *integer division* (in which the fractional part is dis-

carded), and **dow** represents the day of the week, with *Sunday being 0*,

*Monday being 1*, etc.

Write a function called **dayOfWeek** that takes the day, month and year of a

date and returns a number to represent the day of the week for that date,

with *0 for Monday*, *1 for Tuesday* and so on (Note the slight difference

from the outcome of Gauss’s algorithm).

If the numbers do not represent a valid date the function should return

**None**.

7

**e.** Write a function called **printCalendar** that takes two integers to represent

a particular month in a particular year and prints out the calendar for that

month.

For this function, you can assume that the month is between 1 and 12 and

the year is between 1800 and 20000.

**f.** Add statements to your program to ask the user for a birth date of a person

(day, month and year).

If the birth date is not valid, print the message

**The given date is invalid**.

Otherwise, display

• On which day of the week the person was born, e.g.

**The person was born on a Monday**

and

• The calendar of the *month of the birthday* in *2020*. Make sure that the

columns (of the days of the week and the days) line up nicely.

**Q:** Should I submit more than one program for this question?

**A:** No, please put all the functions and the statements required for the last

task in a program.

**Q:** What about test cases?

**A:** You are not required to do this. But it may be a good idea to include the

testing code. But make sure you comment the code for testing out.

\_

8