1811ICT/2807ICT/7001ICT Programming Principles Workshop 9

School of Information and Communication Technology

Griffith University

|  |  |
| --- | --- |
| Goals | This workshop focusses on everything in the course up to files. |
| When | Workshops from Friday 20 May to Thursday 26 May |
| Marks | 3 |
| Due | Pre-workshop questions before the start of the above mentioned workshops  Workshop programming problems by 11:59pm on 29 May |

# Before your workshop class:

* Read all of this document.
* Review the lecture notes sections 1 to 24.
* **Complete the pre-workshop questions (1 mark) posted on the course website and submit the answers for marking**.

# Workshop activities

At any stage, when you are stuck, *ask your tutor*!

## Problem 1

*Problem:* The people of ancient land of Pacific Baza had a simple mathematical system that knew only natural numbers and addition. The genius Bazan scholar, Gringo el Possum, built a computer from wood and various animal parts. Archeologists have recovered ancient scrolls with enough scraps of programs to reconstruct the programming language he named, *Adder*.

The Adder language has only a few simple statements:

|  |  |
| --- | --- |
| quit | Exit the REPL or terminate a program. |
| input *var* | Prompt for and allow the user to enter a value for the variable named *var*. |
| print *val* | Print the value *val*. |
| *var* gets *val* | variable *var* is assigned the value *val*. |
| *var* adds *val* | variable *var* has the value *val* added to it. |

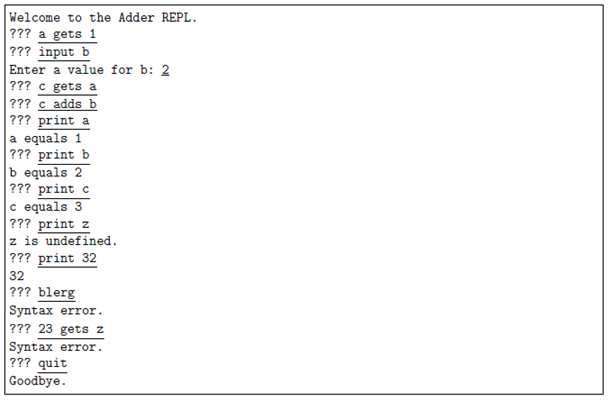
where:

* *var* is always a variable name that contains only letters; and
* *val* can be either:

**–** a variable name that contains only letters; or

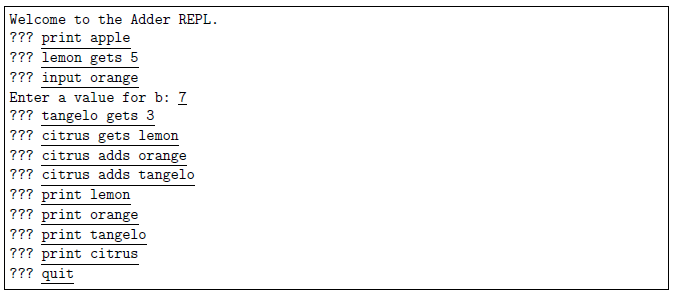
**–** a natural number that contains only digits.

The Adder REPL allows the user to enter commands interactively. For example:



Write the program for the Adder REPL. Hints: Make good use of string methods. Can you divide your program up into smaller pieces by defining functions?

*Answer*: Copy your code in the space given below and insert screenshots of your program output for the following scenario:



***Copy your code here***

# Matthew Prendergast

# 24th May, 2022 - Problem 1 (Workshop - Week 9)

def input\_var(var, val):

    """This function inputs a key and a value to the dictionary"""

    global input\_variable

    input\_variable[var] = int(val)

    return

def input\_print(var):

    """This function prints something to the screen given a certain input"""

    global input\_variable

    if var.isdigit():

        print(var)

    elif var in input\_variable:

        print(f"{var} equals {input\_variable[var]}")

    else:

        print(f"{var} is undefined.")

    return

def input\_gets(var, val):

    """This function assigns variables new values using the get keyword"""

    global input\_variable

    if var.isalpha():

        if val.isdigit():

            input\_var(var, val)

        elif val in input\_variable:

            val\_temp = input\_variable[val]

            input\_var(var, val\_temp)

        else:

            print("Syntax error.")

    else:

        print("Syntax error.")

    return

def input\_adds(var, val):

    """This function adds values to variables using the adds keyword"""

    global input\_variable

    if var.isalpha():

        if val.isdigit():

            input\_variable[var] = input\_variable[var] + int(val)

        elif val in input\_variable:

            val\_temp = input\_variable[val]

            input\_variable[var] = input\_variable[var] + val\_temp

        else:

            print("Syntax error.")

    else:

        print("Syntax error.")

    return

def main():

    """This is the main program code"""

# Initialise the empty dictionary.

input\_variable = {}

# Get the initial input from the user.

print("Welcome to the Adder REPL.")

statement = input("??? ")

statement = statement.strip().lower()

# If the user does not want to quit the program, proceed.

while statement != "quit":

    # Split the statement into a list to be read.

    state\_list = statement.split()

    # If the user wants to input a variable.

    if len(state\_list) == 2 and state\_list[0] == "input":

        var = state\_list[1]

        if var.isalpha():

            val = input(f"Enter a value for {var}: ")

            while not val.isdigit():

                val = input(f"Enter a value for {var}: ")

            input\_var(var, val)

    # If the user wanted to print something to the screen.

    elif len(state\_list) == 2 and state\_list[0] == "print":

        var = state\_list[1]

        input\_print(var)

    # If the user wants to assign a variable with gets keyword.

    elif len(state\_list) == 3 and state\_list[1] == "gets":

        var = state\_list[0]

        val = state\_list[2]

        input\_gets(var, val)

    # If the user wants to add two variables with adds keyword.

    elif len(state\_list) == 3 and state\_list[1] == "adds":

        var = state\_list[0]

        val = state\_list[2]

        input\_adds(var, val)

    else:

        print("Syntax error.")

    # Repromt the user for a new input.

    statement = input("??? ")

    statement = statement.strip().lower()

# Print Goodbye on the exit.

print("Goodbye.")

# This calls the main function.

main()

***Insert your screenshot here***

Text

Description automatically generated

## Problem 2

*Problem:* Write an Adder interpreter, that prompts for and executes an Adder script. For example if the file children.ad contains:

input sons   
input daughters   
children gets sons   
children adds daughters   
print children   
quit

The interpreter would run like this

Script name: children.ad

Enter a value for sons: 3   
Enter a value for daughters: 4   
children equals 7

Hint: This should involve a few small modifications to your REPL from Problem 1.

*Answer*: Copy your code in the space given below and insert screenshots of your program output for the following scenario:

* Use the file P2.ad as the source file.

***Copy your code here***

# Matthew Prendergast

# 24th May, 2022 - Problem 2 (Workshop - Week 9)

def input\_var(var, val):

    """This function inputs a key and a value to the dictionary"""

    global input\_variable

    input\_variable[var] = int(val)

    return

def input\_print(var):

    """This function prints something to the screen given a certain input"""

    global input\_variable

    if var.isdigit():

        print(var)

    elif var in input\_variable:

        print(f"{var} equals {input\_variable[var]}")

    else:

        print(f"{var} is undefined.")

    return

def input\_gets(var, val):

    """This function assigns variables new values using the get keyword"""

    global input\_variable

    if var.isalpha():

        if val.isdigit():

            input\_var(var, val)

        elif val in input\_variable:

            val\_temp = input\_variable[val]

            input\_var(var, val\_temp)

        else:

            print("Syntax error.")

    else:

        print("Syntax error.")

    return

def input\_adds(var, val):

    """This function adds values to variables using the adds keyword"""

    global input\_variable

    if var.isalpha():

        if val.isdigit():

            input\_variable[var] = input\_variable[var] + int(val)

        elif val in input\_variable:

            val\_temp = input\_variable[val]

            input\_variable[var] = input\_variable[var] + val\_temp

        else:

            print("Syntax error.")

    else:

        print("Syntax error.")

    return

def main():

    """This is the main program code"""

# Initialise the empty dictionary.

input\_variable = {}

# Prompt user to input a the file name.

file = input("Script name: ")

# Open the file to read:

while True:

    try:

        fhand = open(file, mode = "r")

        break

    except:

        print("Error: File cannot be opened")

        file = input("Script name: ")

lines = fhand.readlines()

count = 0

# Get the initial input from the user.

statement = lines[count].strip().lower()

# If the user does not want to quit the program, proceed.

while statement != "quit":

    # Split the statement into a list to be read.

    state\_list = statement.split()

    # If the user wants to input a variable.

    if len(state\_list) == 2 and state\_list[0] == "input":

        var = state\_list[1]

        if var.isalpha():

            val = input(f"Enter a value for {var}: ")

            while not val.isdigit():

                val = input(f"Enter a value for {var}: ")

            input\_var(var, val)

    # If the user wanted to print something to the screen.

    elif len(state\_list) == 2 and state\_list[0] == "print":

        var = state\_list[1]

        input\_print(var)

    # If the user wants to assign a variable with gets keyword.

    elif len(state\_list) == 3 and state\_list[1] == "gets":

        var = state\_list[0]

        val = state\_list[2]

        input\_gets(var, val)

    # If the user wants to add two variables with adds keyword.

    elif len(state\_list) == 3 and state\_list[1] == "adds":

        var = state\_list[0]

        val = state\_list[2]

        input\_adds(var, val)

    else:

        print("Syntax error.")

    # Input the next line.

    count +=1

    statement = lines[count].strip().lower()

# Close the file on the exit.

fhand.close()

# This calls the main function.

main()

***Insert your screenshot here***

Text

Description automatically generated

## Problem 3

*Problem:* A road map defines locations as map references like B3, where B is the *x*-coordinate value and 3 is the *y*-coordinate.

A

B

C

D

E

F

Y

Z

1

2

3

4

5

6

25

26

…

⫶

B3

E2

Y25

The grid lines are 0.5 km apart.

Write a program that allows the user to enter a trip as a sequence of any number of map references on one line, and reports the total length of the trip, assuming they can travel in straight lines. For example:

Enter trip map references: C2 B5 Y25

Total distance = 16.8 km

For badly formatted map references, your program should exit, reporting the first bad map reference.

Enter trip map references: E6 E4 D7 d43 F5

Bad reference: d43

Hints: You need to *split* the input line into separate references; each reference starts with one character which must be an upper case letter, and the rest must be only digits; and Pythagoras will help. The function exit() can abort the program if you detect an error in the input.

*Answer*: Copy your code in the space given below and insert screenshots of your program output for the following two scenarios:

* A1 B2 C3 D20 S15 W25 Z26
* D2 F23 Ja E23 Z2

***Copy your code here***

# Matthew Prendergast

# 24th May, 2022 - Problem 3 (Workshop - Week 9)

from math import sqrt

# Prompt the user to input the grid references.

x = input("Enter trip map references: ")

references = x.split()

# Set the global distance for each grid space and initialise the total distance to 0.

GRID = 500

distance = 0

# Check the list for bad references.

for j in range(len(references)):

    if not references[j][0].isupper():

        print("Bad reference:", references[j])

        exit()

    elif not references[j][1:].isdigit():

        print("Bad reference:", references[j])

        exit()

# Calculate the distance between each position and increment the distance count.

for i in range(len(references) - 1):

    y1 = ord(references[i][0])

    x1 = int(references[i][1:])

    y2 = ord(references[i + 1][0])

    x2 = int(references[i + 1][1:])

    distance += sqrt(abs(y1 - y2) \*\* 2 + abs(x1 - x2) \*\* 2) \* GRID

# Print the result.

print(f"Total distance = {distance / 1000:.1f} km")

***Insert your screenshot here***

Text

Description automatically generated

Text

Description automatically generated

# Submission and marking

The pre-workshop can be accessed and submitted online using the provided link in the course website. Students get 1 mark if they get >50% in pre-workshop questions, or 0.5 mark if they get 0%-50% in pre-workshop questions, or 0 marks without any attempt.

For workshop tasks, please submit this document with copied codes and inserted screenshots using the provided submission link in the course website. Students get 2 marks if they complete two or more problems correctly, or 1 mark if they complete one problem correctly, or 0 marks without any attempt.