Beijing National Day School
Department of Mathematics & Computer Science

AP Computer Science Principles

Semester 1 Exam

Exam Record	
Multiple Choice	/ 50 pts
Short Answer	/ 30 pts
Programming	/ 10 pts
Total:	/ 90 pts
Grade:	

English Name: _____

Pinyin Name: _____

Mr. Alwin Tareen, January 2020

Part I: Python Multiple Choice (50 points)

- Number of Python multiple choice questions: 50. Percent of total grade: 56%.
- Determine the answer to each of the following questions, using the available space for any necessary scratchwork.
- Decide which is the best of the choices given, and select the correct answer by filling in the corresponding bubble on the separate answer sheet.

(1pt) **1.** Which of the following choices is a legal and legitimate `Python` variable name?

1 pt

A `2bad4you`
B `calvin&hobbes`
C `year2000`
D `#hammertime`

(1pt) **2.** You would like to set up a variable called `ounces` that has the value `16`. What simple `Python` statement will accomplish this?

1 pt

A `ounces = 16`
B `16 = ounces`
C `def ounces(16):`
D `ounces(16)`

(1pt) **3.** What does the following `Python` statement print out:
`print("123" + "abc")`

1 pt

A `"123" + "abc"`
B This is a syntax error because you cannot add strings.
C `123+abc`
D `123abc`

(1pt) **4.** In `Python`, the `float` data type is used to store:

1 pt

A booleans
B decimal numbers
C strings
D integers

(1pt) **5.** What is the result of the following `Python` statement:
`print(42%10)`

1 pt

A `1042`
B `420`
C `4`
D `2`

5 pts

(1pt) **6.** Which of the following choices is the correct assignment statement for a `string` data type?

1 pt

A `greetings = [Hello]`
B `greetings = @Hello@`
C `greetings = "Hello"`
D `greetings = #Hello#`

(1pt) **7.** What is the result of the following `Python` statement:
`print(17/4)`

1 pt

A `4`
B `4.0`
C `4.3`
D `4.25`

(1pt) **8.** What are the only values that are permissible in `Python`'s `boolean` data type?

1 pt

A `Yes, No`
B `On, Off`
C `Right, Wrong`
D `True, False`

(1pt) **9.** Which of the following is a comment in `Python`?

1 pt

A `/* This is a test */`
B `// This is a test`
C `# This is a test`
D `% This is a test`

(1pt) **10.** Which of the following elements of a mathematical expression in `Python` is evaluated first?

1 pt

A Multiplication `*`
B Addition `+`
C Parenthesis `()`
D Subtraction `-`

(1pt) **11.** What will be the value of `x` when the following statement is executed: `x = int(98.6)`

1 pt

A `99`
B `6`
C `98`
D `100`

(1pt) **12.** What does the `Python` function `input()` do?

1 pt

A Pause the program and read data from the user.
B Take a screen shot from an area of the screen.
C Read the memory of the running program.
D Connect to the network and retrieve a web page.

7 pts

(1pt) **13.** Which `Python` keyword indicates the start of a function definition?

A `sweet`
B `def`
C `continue`
D `return`

1 pt

(1pt) **14.** Consider the following function definition:
```
def circlearea(radius):
```
In this context, what is the formal name for the variable `radius`?

A expression
B logical deduction
C parameter
D condition

1 pt

(1pt) **15.** Which of the following is NOT a valid string method in `Python`?

A `boldface()`
B `startswith()`
C `upper()`
D `strip()`

1 pt

(1pt) **16.** What does the following Python program print out?
```
str1 = "Hello"
str2 = "there"
greet = str1 + str2
print(greet)
```
A `Hello there`
B `Hellothere`
C `there`
D `Hello`

1 pt

(1pt) **17.** How would you use the index operator to print out the letter `"q"` from the following string?
```
x = "From marquard@uct.ac.za"
```
A `print(x[9])`
B `print(x[8])`
C `print(x[-1])`
D `print(x[q])`

1 pt

(1pt) **18.** How would you use string slicing to print out `"uct"` from the following string?
```
x = "From marquard@uct.ac.za"
```
A `print(x[14+17])`
B `print(x[15:18])`
C `print(x[14:17])`
D `print(x[14:3])`

1 pt

6 pts

(1pt) **19.** What is the iteration variable in the following `Python` code?

```
for letter in "banana":
    print(letter)
```

A `letter`
B `print`
C `in`
D `"banana"`

1 pt

(1pt) **20.** How would you print out the following string in all upper case in `Python`?

```
greet = "Hello there"
```

A `puts greet.ucase;`
B `print(uc($greet))`
C `print(greet.upper())`
D `console.log(greet.toUpperCase());`

1 pt

(1pt) **21.** What does the following Python program print out?

```
data = "From stephen.marquard@uct.ac.za"
pos = data.find(".")
print(data[pos:pos+3])
```

A `uct`
B `mar`
C `.ma`
D `ste`

1 pt

(1pt) **22.** Consider the following string declaration:

```
grocery = "Mango"
```

Which of the following statements would cause an error(also known as a traceback)?

A `dance = "T" + grocery[1:]`
B `person = grocery[:-2]`
C `several = grocery * 3`
D `grocery[0] = "T"`

1 pt

(1pt) **23.** Consider the following Python code:

```
lunch = "pizza"
dinner = lunch[:]
```

Note that the `start` and `stop` indexes are omitted from the square bracket notation. What is the technical term for the outcome of this kind of string slicing?

A concatenation
B immutable
C clone
D iteration

1 pt

5 pts

(1pt) **24.** Consider the text character `"a"`. How would you determine its corresponding numerical `ASCII` code?

1 pt

A Use the `ord()` function as follows: `num = ord("a")`
B Apply the `chr()` function, such as: `val = chr("a")`
C Generate the `ASCII` value with: `code = ascii("a")`
D Convert the `"a"` to an integer value: `out = int("a")`

(1pt) **25.** Consider the following string: `sport = "BASKETBALL"`. How would you convert this string to lowercase?

1 pt

A `output = sport.shrink()`
B `solution = sport.smaller()`
C `outcome = lowercase(sport)`
D `result = sport.lower()`

(1pt) **26.** What would be the output of the following Python code:

```
meal = "cheese and mushroom pizza delivered to your dorm room."
result = meal.find("room")
print(result)
```

1 pt

A `14`
B `15`
C `49`
D `16`

(1pt) **27.** For the following list, how would you print out `"Sally"`?

```
friends = ["Joseph", "Glenn", "Sally"]
```

1 pt

A `print(friends[3])`
B `print(friends["Sally"])`
C `print(friends[2])`
D `print(friends[2:1])`

(1pt) **28.** Which of the following Python statements would print out the length of a list stored in the variable `fruit`?

1 pt

A `print(length(fruit))`
B `print(fruit.length())`
C `print(len(fruit))`
D `print(strlen(fruit))`

(1pt) **29.** What type of data is produced when you call the `range()` function? For example, consider the statement: `nums = range(5)`

1 pt

A A list of characters
B A list of integers
C A list of words
D A string

6 pts

(1pt) **30.** What does the following Python code print out?

```
first = [1, 2, 3]
second = [4, 5, 6]
nums = first + second
print(len(nums))
```

A `[1, 2, 3]`
B `[1, 2, 3, 4, 5, 6]`
C `[4, 5, 6]`
D `6`

1 pt

(1pt) **31.** Which of the following slicing operations will produce the list `[12, 3]`?

```
nums = [9, 41, 12, 3, 74, 15]
```

A `nums[1:3]`
B `nums[2:4]`
C `nums[2:2]`
D `nums[12:3]`

1 pt

(1pt) **32.** Which list method adds a new item to the end of an existing list?

A `add()`
B `append()`
C `index()`
D `push()`

1 pt

(1pt) **33.** What will the following Python code print out?

```
friends = ["Joseph", "Glenn", "Sally"]
friends.sort()
print(friends[0])
```

A `Glenn`
B `Joseph`
C `friends`
D `Sally`

1 pt

(1pt) **34.** Which of the following Python functions deletes an element from a list?

A `push()`
B `pop()`
C `invalidate()`
D `split()`

1 pt

(1pt) **35.** Which of the following Python functions breaks a string into a list of words?

A `split()`
B `join()`
C `remove()`
D `extend()`

1 pt

6 pts

(1pt) **36.** What task does the following Python code perform?

```
for num in range(1, 10, 2):
    print(num)
```

A It prints all the ODD numbers in the range [1, 9]
B It prints all numbers in the range[1, 9]
C This code fails with a traceback.
D It prints all the EVEN numbers in the range [1, 10]

1 pt

(1pt) **37.** What is the purpose of the second parameter of the `get()` method for Python dictionaries?

A It signifies a key which must be placed in the dictionary.
B It specifies a unique key that the programmer wishes to retrieve.
C It indicates the particular value that the programmer wants to retrieve.
D To provide a default value if the key(from the first parameter of the `get()` method) does not exist in the dictionary.

1 pt

(1pt) **38.** How are Python dictionaries different from Python lists?

A Python lists can store multiple values, whereas Python dictionaries store a single value.
B Python lists can store strings, while Python dictionaries can only store words.
C Python lists are indexed using integers, whereas Python dictionaries are indexed with any immutable data type.
D Python dictionaries are mutable, while Python lists are immutable.

1 pt

(1pt) **39.** What would be the output produced by the following Python code?

```
fruit = {"banana":5, "pear":3, "orange":8}
result = fruit["kiwi"]
print(result)
```

A `0`
B This program would fail with a traceback.
C `kiwi`
D `-1`

1 pt

(1pt) **40.** What would be the output produced by the following Python code?

```
fruit = {"banana":5, "pear":3, "orange":8}
result = fruit.get("kiwi", 0)
print(result)
```

A `0`
B This program would fail with a traceback.
C `kiwi`
D `-1`

1 pt

5 pts

(1pt) **41.** Consider the following Python code, in which we loop through a dictionary. What are the items that the `for` loop iterates through?

```
fruit = {"banana":5, "pear":3, "orange":8}
for item in fruit:
    print(item)
```

A The keys in the dictionary.
B The integers in `range(0, len(fruit))`
C The values in the dictionary.
D All of the mutable data types in the dictionary.

1 pt

(1pt) **42.** Which of the following Python methods would you use to create a separate and distinct copy of a dictionary?

A `double()`
B `duplicate()`
C `copy()`
D `clone()`

1 pt

(1pt) **43.** Consider the following Python dictionary:

```
fruit = {"banana":5, "pear":3, "orange":8}
```

Which of the following statements would correctly remove the key-value pair `"orange":8` from this dictionary?

A `remove.fruit["orange"]`
B `eliminate("orange":8)`
C `del fruit[8]`
D `del fruit["orange"]`

1 pt

(1pt) **44.** What would be the output produced by the following Python code?

```
drinks = {"coffee":87, "tea":23, "juice":49}
result = drinks.values()
print(result)
```

A `(78, 32, 94)`
B `[("coffee",87), ("tea",23), ("juice",49)]`
C `["coffee", "tea", "juice"]`
D `[87, 23, 49]`

1 pt

(1pt) **45.** Consider the following Python dictionary:

```
fruit = {"banana":5, "pear":3, "orange":8}
```

Which of the following Python statements would correctly subtract `2` from the value corresponding to the key `"orange"`?

A `fruit["orange"].reduce(2)`
B `orange subtraction 2`
C `fruit.orange.minus.2`
D `fruit["orange"] -= 2`

1 pt

5 pts

(1pt) **46.** Consider the following Python dictionary:
```
cheese = {"swiss":3, "cheddar":7, "gouda":6}
```
Which of the following Python statements indicates whether `"swiss"` appears as a key in the dictionary `cheese`?

1 pt

A `cheese.excludevalue("cheddar", "gouda")`
B `"swiss" in cheese`
C `cheese.containsvalue("swiss")`
D `cheese --> "swiss"`

(1pt) **47.** Consider the following Python list:
```
toppings = ["cheese", "pepperoni"]
```
How would we alter this list so that the following items are concatenated onto the end:
```
extras = ["mushroom", "sausage", "ham"]
```

1 pt

A `toppings.push(extras)`
B `toppings.extend(extras)`
C `toppings.concatenate(extras)`
D `toppings.tack(extras)`

(1pt) **48.** Consider the following Python list:
```
breakfast = ["eggs", "juice", "toast"]
```
How would you place the element `"bacon"` at the beginning of this list?

1 pt

A `breakfast.insert(0, "bacon")`
B `breakfast.insert(3, "bacon")`
C `breakfast.push("bacon", 0)`
D `breakfast.place(0, "bacon")`

(1pt) **49.** Consider the following Python list:
```
snacks = ["pizza", "wings", "soda", "chips"]
```
How would you remove the element `"soda"` from this list?

1 pt

A `snacks.remove(2)`
B `snacks.eliminate("soda")`
C `snacks.destroy(2)`
D `snacks.remove("soda")`

(1pt) **50.** Consider the following Python list:
```
dinner = ["salad", "tea", "juice", "soda", "hamburger", "pizza"]
```
Let's say that you wanted to remove all of the **drinks** from this list, leaving just the following items:
```
["salad", "hamburger", "pizza"]
```
Which of the following choices would accomplish this?

1 pt

A `del dinner[2:5]`
B `remove dinner[1:4]`
C `del dinner[1:4]`
D `dinner.remove(drinks)`

5 pts

Part II: Python Short Answer (30 points)

- Number of Python short answer questions: 30. Percent of total grade: 33%.
- Solve each of the following short answer questions. Write your solution in the corresponding box labelled, "Answer:".

(1pt) **1.** What is the output of the following `Python` code:

```
print(3 > 4 or (2 < 3 and 9 > 10))
```

Answer:

1 pt

(1pt) **2.** What is the output of the following `Python` code:

```
lunch = "cheeseburgers"
print(lunch[6:12])
```

Answer:

1 pt

(1pt) **3.** What is the output of the following `Python` code:

```
breakfast = "pineapple"
print(breakfast[:4])
```

Answer:

1 pt

(1pt) **4.** What is the output of the following `Python` code:

```
flavor = "strawberry"
print(flavor[5:])
```

Answer:

1 pt

(1pt) **5.** What is the output of the following `Python` code:

```
icecream = "vanilla"
print(icecream[:])
```

Answer:

1 pt

(1pt) **6.** What is the output of the following `Python` code:

```
drink = "soda"
print(drink[:-1])
```

Answer:

1 pt

(1pt) **7.** What is the output of the following `Python` code:

```
beverage = "water"
print(beverage * 3)
```

Answer:

1 pt

(1pt) **8.** What is the output of the following `Python` code:

```
greetings = "Hello, world!"
newgreetings = "J" + greetings[1:]
print(newgreetings)
```

Answer:

1 pt

8 pts

(1pt) **9.** What is the output of the following `Python` code:

```
print("cola" in "chocolate")
```

Answer:

1 pt

(1pt) **10.** What is the output of the following `Python` code:

```
fruit = "kiwi"
bigfruit = fruit.upper()
print(bigfruit)
```

Answer:

1 pt

(1pt) **11.** What is the output of the following `Python` code:

```
citrus = "ORANGE"
smallcitrus = citrus.lower()
print(smallcitrus)
```

Answer:

1 pt

(1pt) **12.** What is the output of the following `Python` code:

```
vegetable = "cauliflower"
index = vegetable.find("u")
print(index)
```

Answer:

1 pt

(1pt) **13.** What is the output of the following `Python` code:

```
line = "Please have a nice day"
print(line.startswith("Please"))
```

Answer:

1 pt

(1pt) **14.** What is the output of the following `Python` code:

```
meal = "fresh pizza is the best pizza"
print(meal.replace("pizza", "salad"))
```

Answer:

1 pt

(1pt) **15.** What is the output of the following `Python` code:

```
def choose(x, y, z):
    if x:
        return y
    else:
        return z
print(choose(False, 2, 3))
```

Answer:

1 pt

(1pt) **16.** What is the output of the following code:

```
cheeses = ["Cheddar", "Edam", "Gouda"]
print(cheeses[0])
```

Answer:

1 pt

8 pts

(1pt) **17.** What is the output of the following code:

```
print([0] * 4)
```

Answer:

1 pt

(1pt) **18.** What is the output of the following code:

```
snacks = ["pizza", "burger"]
snacks.append("fries")
print(snacks)
```

Answer:

1 pt

(1pt) **19.** What is the output of the following code:

```
drinks = ["tea", "soda", "cola", "juice"]
drinks.sort()
print(drinks)
```

Answer:

1 pt

(1pt) **20.** What is the output of the following code:

```
dinner = ["salad", "bread", "steak", "potato"]
del dinner[1]
print(dinner)
```

Answer:

1 pt

(1pt) **21.** What is the output of the following code:

```
nums = [3, 41, 12, 9, 74, 15]
print(max(nums))
```

Answer:

1 pt

(1pt) **22.** What is the output of the following code:

```
food = {"pizza":3}
food["fries"] = 10
print(food)
```

Answer:

1 pt

(1pt) **23.** What is the output of the following code:

```
treasure = {"gold":50, "silver":100}
print("gold" in treasure)
```

Answer:

1 pt

(1pt) **24.** What is the output of the following code:

```
inventory = {
    "pocket":"lint",
    "canteen":"water",
    "pouch":"flint",
    "backpack":["shovel", "bedroll", "rope"]
}
print(inventory["backpack"])
```

Answer:

1 pt

8 pts

(1pt) **25.** What is the output of the following code:

```
fortune = {"gold":500}
fortune["gold"] += 50
print(fortune)
```

Answer:

1 pt

(1pt) **26.** What is the output of the following code:

```
inventory = {
    "gold":500,
    "backpack":["xylophone", "dagger", "bedroll"]
}
inventory["backpack"].sort()
print(inventory["backpack"])
```

Answer:

1 pt

(1pt) **27.** What is the output of the following code:

```
grocery = {"kiwi":5, "grape":12}
del grocery["kiwi"]
print(grocery)
```

Answer:

1 pt

(1pt) **28.** Consider the following dictionary:

```
salad = {"caesar":1, "garden":2}
```

Write an assignment statement that modifies this dictionary to become the following:

```
salad = {"caesar":1, "garden":2, "vegetable":3}
```

Answer:

1 pt

(1pt) **29.** What is the output of the following code:

```
singer = {"justin":"bieber", "taylor":"swift", "ed":"sheeran"}
print(singer.get("swift", "guitar"))
```

Answer:

1 pt

(1pt) **30.** What is the output of the following code:

```
sports = {"tennis":43, "football":78, "badminton":52}
result = list(sports.keys())
print(result)
```

Answer:

1 pt

6 pts

Part III: Python Programming (10 points)

- Number of Python programming questions: 4. Percent of total grade: 11%.
- Show all of your work. Remember that program segments are to be written in the `Python` programming language.
- In writing solutions for each question, you may use any of the accessible methods that are listed in the **Python Quick Reference** sheet.

(2pts) **1. Background Theory: The DNA Molecule**

2 pts

The following are some facts regarding the double helix structure of DNA.

- The DNA molecule is made up of two **strands**, running in opposite directions.
- Each base bonds to a base in the opposite strand. *Adenine* always bonds with *Thymine*, and *Cytosine* always bonds with *Guanine*. The **complement** of a base is the opposite base to which it always bonds.
- The two strands are twisted together into a long spiral staircase structure called a **double helix**.
- If we know the order of bases on one strand, we can immediately deduce the sequence of bases in the complementary strand. These bases will run in the opposite order, to match the fact that the two strands of DNA run in opposite directions.
- In DNA strings, the bases *Adenine* and *Thymine* are complements of each other, and they are represented by the letters A and T respectively.
- Also, the bases *Cytosine* and *Guanine* are complements of each other, and they are represented by the letters C and G.

Specification

Write a `Python` function that takes in a string of DNA base symbols as a parameter, and returns the complementary strand that corresponds to it. You will have to reverse the order of the symbols in the DNA string, then find the complement base of each of those symbols. Your function should be called `reversecomplement(dna)`, which takes in a single parameter, `dna`.

The function should return a `string`.

Hint: You may find it useful to create a dictionary consisting of the DNA bases as keys, and the complements as values.

Testing

- If the following statements are executed:

```
result = reversecomplement("AAAACCCGGT")
print(result)
```

- Then the output of your program should be:

```
ACCGGGTTTT
```

Write your answer on the next page.

2 pts

```
def reversecomplement(dna):
    # YOUR CODE HERE
```

0 pts

(2pts) **2. Background Theory: Rocket Science**

2 pts

SpaceX is a private aerospace manufacturer and space transportation services company that was founded in 2002 by Elon Musk. One of its more successful projects was the Falcon launch vehicle, a rocket ship that has the unique ability to land in an upright stance.

In order to launch the Falcon rocket ship into orbit, engineers must first determine the amount of fuel that is required, and this depends upon the rocket's **mass**. Let's say that the fuel equation is as follows: take the mass of the Falcon rocket, divide by three, discard the decimal component of that result, and then subtract two, to get your answer.

For example:

- Consider the case where the Falcon rocket has a mass of 23. Dividing this by 3 yields a result of 7.66666. Then, discard the decimal component to get 7. Subtracting 2 from this results in a final answer of 5.
- **Hint:** Use the `math.floor(num)` function to discard the decimal component of `num`. It is a member of the `math` library.

The Problem Scope

Unfortunately, when it comes to rocket science, things aren't so simple. Fuel itself has a mass component, and thus, it also requires fuel to be launched into orbit. Luckily, we can use our previously mentioned equation for this: take the fuel's mass, divide by 3, discard the decimal component of the result, and subtract 2. However, that fuel **also** requires fuel, and **that** fuel requires fuel, and so on. Note that any mass that would require **negative fuel** should instead be treated as if it requires **zero fuel**.

So, for the mass of the Falcon rocket, calculate its required fuel and add it to the total. Then, treat the fuel amount that you just calculated as input mass, and repeat the process, continuing until a fuel requirement is zero or negative.

For example:

- At first, a Falcon rocket of mass 1969 requires 654 fuel. Then, this fuel requires 216 more fuel($654/3 - 2$). 216 then requires 70 more fuel, which requires 21 more fuel, which requires 5 fuel, which requires no further fuel. So, the total fuel required for a Falcon rocket of mass 1969 is: $654 + 216 + 70 + 21 + 5 = 966$.

Specification

Write a `Python` function that takes in a `float` value `mass` as a parameter, representing the mass of the Falcon rocket. Then, it calculates the amount of fuel required to launch the rocket into orbit, according to the previously mentioned requirements.

The function should return a `float`. Assume that the `math` library has been imported.

Write your solution on the next page.

2 pts

Testing

If the following statements are executed:

```
result = rocketFuel(1969.0);
print(result);
```

Then the output of your program should be: `966.0`

```
import math
def rocketFuel(mass):
    # YOUR CODE HERE
```

0 pts

(3pts) **3. Background Theory: The One-Time Pad**

3 pts

The One-Time Pad is an encryption technique that is similar to the Caesar Cipher, and has been proven to be unbreakable. The key idea is to shift each character in a plaintext message by some random amount between `1` and `26`, inclusive. If this shift amount is large enough to go beyond the end of the alphabet, then it "wraps around" and continues from the beginning.

For example, let's say we want to shift the letter `y` by `5` positions. Obviously, a shift of `1` takes us to `z`, which is the end of the alphabet. Therefore, we resume shifting from the beginning of the alphabet, starting from `a`. In this manner, shifting `y` by `5` positions will result in the letter `d`.

An easy way to implement this shifting behavior is to first convert the text character into its corresponding `ASCII` value, using the `ord()` function. For example, the character `"a"` has the `ASCII` value of `97`. However, in order to have the wraparound effect, the letters `a...z` must correspond to the numerical range `0...25`. Subtracting an offset value of `97` from the resulting `ASCII` value will achieve this.

The Problem Scope

In general, if p is the `ASCII` value of a letter in the plaintext, and k is the amount by which each letter is shifted, then the `ASCII` value of the corresponding letter in the ciphertext c, is computed by the following equation:

$$c \quad = \quad (p + k) \,\%\, 26$$

Note that each character in a One-Time Pad is shifted by its own individual amount. Therefore, there must be a corresponding numerical list which contains as many shift amounts as there are letters in the plaintext message. For example, let's say the plaintext message is `secret`, and the shift amounts are `[1, 3, 2, 10, 8, 2]`. The encrypted ciphertext would be: `thebmv`.

Specification

Write a `Python` function called `onetimepad(plaintext, shifts)` that takes in a `plaintext` message and a list of integers called `shifts` as parameters, and returns the encrypted version of this message by using the One-Time Pad. Assume that the `plaintext` message only consists of lowercase text characters.

The function should return a `string`.

Testing

If the following statements are executed:

```
result = onetimepad("secret", [1, 3, 2, 10, 8, 2])
print(result)
```

Then the output of your program should be: `thebmv`

3 pts

```
def onetimepad(plaintext, shifts):
    # YOUR CODE HERE
```

0 pts

(3pts) **4. Background Theory: Machine Readable Passports**

3 pts

The International Civil Aviation Organization, a Quebec-based group, developed standards for Machine Readable Travel Documents(MRTDs), which includes passports and visas. Machine Readable Travel Documents have a section known as the Machine Readable Zone(MRZ).

The MRZ consists of two lines of text characters, and each line is exactly 44 characters long. No blank spaces are permitted in the MRZ, so the individual data parts are separated by the less than sign(<). The following is a sample MRZ of a person named Anna Maria Eriksson, from the fictional land of Utopia.

```
P<UTOERIKSSON<<ANNA<MARIA<<<<<<<<<<<<<<<<<<<<
L898902C<3UTO6908061F9406236ZE184226B<<<<<14
```

The part of the MRZ we will focus on is the 14-character alphanumeric digit on the second line, called the **personal number**. In this case, Anna's personal number is: `ZE184226B<<<<<`

The Problem Scope

Given a particular person number, we want to verify that it is valid by calculating its **check digit**. The procedure is as follows:

- All of the non-numeric characters in the personal number must be converted to numeric digits with the following lookup table. Note that the < character is converted to 0.

letter	<	A	B	C	D	E	F	G	H	I	J	K	L	M
value	0	10	11	12	13	14	15	16	17	18	19	20	21	22

letter	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
value	23	24	25	26	27	28	29	30	31	32	33	34	35

- Each individual number in the result must be multiplied by a corresponding weight value. The first number is multiplied by 7, the second number is multiplied by 3, and the third number is multiplied by 1. This 7, 3, 1 pattern then repeats from the fourth number onwards.
- Each of these partial products are added together, and we perform a modulus 10 on this sum. The result is the check digit.

Example Case

- For example, let's consider the personal number `AB2134<<<<<<<<`. If we convert each of the non-numeric characters into numbers, we get the following:

  `[10, 11, 2, 1, 3, 4, 0, 0, 0, 0, 0, 0, 0, 0]`

- Then, let's go ahead and multiply each of these numbers by their corresponding weights, that is, the 7, 3, 1 pattern. This results in the following:

  `[70, 33, 2, 7, 9, 4, 0, 0, 0, 0, 0, 0, 0, 0]`

- Next, sum up all these partial products:

$$70 + 33 + 2 + 7 + 9 + 4 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = 125$$

- Finally, take this result and perform a modulus with 10, that is: $125\%10 = 5$. Therefore, the check digit is `5`.

3 pts

Hints

- A dictionary consisting of each alphabetical character mapped to its corresponding value has been provided for you.
- Also, since the personal numbers are always 14 characters long, a list of the 14-digit 7, 3, 1 weighting pattern has also been provided.
- You will probably need to loop across each character in the personal number and decide whether it is a number or not. Perhaps the `isdigit()` method would be useful here.

Specification

Write a `Python` function called `verifypassport(personal)` that takes in a personal number as a `string`, and calculates the check digit for that personal number.

The function should return a `integer`.

Testing

If the following statements are executed:

```
result = verifypassport("AB2134<<<<<<<<")
print(result)
```

Then the output of your program should be: `5`

Write your solution on the next page.

0 pts

```
def verifypassport(personal):
    mapping = {"<":0, "A":10, "B":11, "C":12, "D":13, "E":14, "F":15, "G":16, "H":17,
    "I":18, "J":19, "K":20, "L":21, "M":22, "N":23, "O":24, "P":25, "Q":26, "R":27,
    "S":28, "T":29, "U":30, "V":31, "W":32, "X":33, "Y":34, "Z":35}
    weights = [7,3,1,7,3,1,7,3,1,7,3,1,7,3]
    # YOUR CODE HERE
```

0 pts

This page is left intentionally blank.

0 pts