MIT-6.00.1x-Introduction-to-Computer-Science-and-Programming-Using-Python-MIDTERM EXAM

Problem 1:

a)Suppose x = "pi" and y = "pie". The line of code x, y = y, x will swap the values of x and y, resulting in x = "pie" and y = "pi".

True

False

b) Suppose x is an integer in the following code:

```
def f(x):
    while x > 3:
        f(x+1)
```

For any value of x, all calls to f are guaranteed to never terminate.

True

False

c)A Python program always executes every line of code written at least once.

True

False

d) #Suppose you have two different functions that each assign a variable called x.

Modifying x in one function means you always modify x in the other function for any x.

True

False

e) The following code will enter an infinite loop for all values of i and j.

```
while i >= 0:
    while j >= 0:
        print(i, j)
```

True

False

f)It is always possible and feasible for a programmer to come up with test cases that run through every possible path in a program.

True

False

g) Assume f() is defined. In the statement a = f(), a is always a function.

True

False

h)A program that keeps running and does not stop is an example of a syntax error.

True

False

I)Consider the following function.

```
def f(x):
   a = []
   while x > 0:
      a.append(x)
      f(x-1)
```

A new object of type list is created for each recursive invocation of f.

True

False

J) A tuple can contain a list as an element.

True

False

Problem 2:

a)Consider the statement: L = {'1':1, '2':2, '3':3}. Which is correct?

1. L is a list
2. L is immutable
3. L contains 6 elements
4. L has integer keys
5. L maps strings to integers

b)Assume a break statement is executed inside a loop and that the loop

1. is inside a function. Which of the following is correct?
2. is inside a function. Which of the following is correct?
3. The program might immediately terminate.
4. The function might immediately terminate.
5. The loop will always immediately terminate.
6. All of the above.
7. None of the above.

c In Python, which of the following is a mutable object?

1. a string
2. a tuple
3. a list
4. all of the above
5. none of the above

d) Assume the statement s[1024] = 3 does not produce an error message.

1. This implies:
2. type(s) can be str
3. type(s) can be tuple
4. type(s) can be list
5. All of the above

e) Consider the code:

```
L = [1,2,3]

d = {'a': 'b'}

def f(x):

   return 3
```

Which of the following does NOT cause an exception (error) to be thrown?

1. Which of the following does NOT cause an exception (error) to be thrown?
2. print(L[3])
3. print(d['b'])
4. for i in range(100001, -1, -2):
5. print(f)
6. print(int('abc'))
7. None of the above

Problem 3

a)Examine the following code snippet:

```
stuff  = _____

for thing in stuff:

   if thing == 'iQ':

      print("Found it")
```

Select all the values of the variable "stuff" that will make the

code print "Found it

- **["iBoy", "iGirl", "iQ", "iC","iPaid","iPad"]**
- **("iBoy", "iGirl", "iQ", "iC","iPaid","iPad")**
- **[("iBoy", "iGirl", "iQ", "iC","iPaid","iPad")]**
- **(["iBoy", "iGirl", "iQ", "iC","iPaid","iPad"],)**
- **["iQ"]**
- **"iQ"**

b) The following Python code is supposed to compute the square of an integer by using successive additions.

```
def Square(x):
   return SquareHelper(abs(x), abs(x))


def SquareHelper(n, x):
   if n == 0:
      return 0
   return SquareHelper(n-1, x) + x
```

Not considering recursion depth limitations, what is wrong with this implementation of procedure Square? Check all that apply.

- It is going to return a wrong value.
- The term Square is a reserved Python keyword.
- Function names cannot start with a capital letter.
- The function is never going to return anything.
- Python has arbitrary precision arithmetic.
- This function will not work for negative numbers.
- The call SquareHelper(abs(x), abs(x)) won't work because you can't have abs(x) as both parameters.
- Nothing is wrong; the code is fine as-is

Problem 4

Write a function is_triangular that meets the specification below.

A triangular number is a number obtained by the continued summation of integers starting from 1. For example, 1, 1+2, 1+2+3, 1+2+3+4, etc.,

# corresponding to 1, 3, 6, 10, etc., are triangular numbers.

```
def is_triangular(k):
     """
    k, a positive integer
   returns True if k is triangular and False if not
    """
#    #YOUR CODE HERE
```

Paste your entire function, including the definition, in the box below.

Do not leave any debugging print statements.

Problem 5

Write a Python function that takes in a string and prints out a version of this string that does not contain any vowels, according to the specification below. Vowels are uppercase and lowercase ('a', 'e', 'i', 'o', 'u'.)

For example, if s = "This is great!" then print_without_vowels will print Ths s grt!. If s = "a" then print_without_vowels will print the empty string .

```
def print_without_vowels(s):
    '''
    s: the string to convert
    Finds a version of s without vowels and whose characters appear in the
    same order they appear in s. Prints this version of s.
    Does not return anything
    '''
    # Your code here
```

Problem 6

Write a function that satisfies the following docstring:

```
def largest_odd_times(L):
    """ Assumes L is a non-empty list of ints
        Returns the largest element of L that occurs an odd number
        of times in L. If no such element exists, returns None """
    # Your code here
```

For example, if:

```
 largest_odd_times([2,2,4,4]) returns None
 largest_odd_times([3,9,5,3,5,3]) returns 9
```

# Paste your entire function, including the definition, in the box below. Do not leave any debugging print statements.

Problem 7

Write a function called dict_invert that takes in a dictionary with immutable values and returns the inverse of the dictionary. The inverse of a dictionary d is another dictionary whose keys are the unique dictionary values in d.

The value for a key in the inverse dictionary is a sorted list (increasing order) of all keys in d that have the same value in d.

Here are two examples:

If d = {1:10, 2:20, 3:30} then dict_invert(d) returns {10: [1], 20: [2], 30:

[3]} If d = {1:10, 2:20, 3:30, 4:30} then dict_invert(d) returns {10: [1], 20:

[2], 30: [3, 4]} If d = {4:True, 2:True, 0:True} then dict_invert(d) returns

{True: [0, 2, 4]}

```
def dict_invert(d):

    '''

    d: dict

    Returns an inverted dictionary according to the instructions above

    '''

    #YOUR CODE HERE
```

Paste your entire function, including the definition, in the box below. Do not

leave any debugging print statements.

Paste your function here

Problem 8

Write a function called general_poly, that meets the specifications below. For example, general_poly([1, 2, 3, 4])(10) should evaluate to 1234 because 1*10^3+2*10^2+3*10^1+4*10^0.

```
def general_poly (L):
            """
            L, a list of numbers (n0, n1, n2, ... nk)
            Returns a  function, which when applied to a value x,
           returns the value n0 * x^k + n1 * x^(k-1) + ... nk * x^0
           """
            #YOUR CODE HERE
```

Problem 9

Write a Python function that takes in two lists and calculates whether they are permutations of each other. The lists can contain both integers and strings.

We define a permutation as follows:

the lists have the same number of elements list elements appear the same

number of times in both lists If the lists are not permutations of each other,

the function returns False.

If they are permutations of each other, the function returns a tuple consisting of the following elements:

the element occuring the most times how many times that element occurs the

type of the element that occurs the most times If both lists are empty return

the tuple (None, None, None). If more than one element occurs the most number

of times, you can return any of them.

```
def is_list_permutation(L1, L2):
    '''
    L1 and L2: lists containing integers and strings
    Returns False if L1 and L2 are not permutations of each other.
        If they are permutations of each other, returns a
        tuple of 3 items in this order:
        the element occurring most, how many times it occurs, and its type
   '''
  # Your code here
```

#For example,

if L1 = ['a', 'a', 'b'] and L2 = ['a', 'b'] then is_list_permutation returns

False if L1 = [1, 'b', 1, 'c', 'c', 1] and L2 = ['c', 1, 'b', 1, 1, 'c'] then

is_list_permutation returns (1, 3, <class 'int'>) because the integer 1 occurs

the most, 3 times, and the type of 1 is an integer (note that the third

element in the tuple is not a string). Paste your entire function, including

the definition, in the box below. Do not leave any debugging print statements.

END OF EXAM