

CSc 110 Sample Midterm Exam #2

1. While Loop Simulation

For each call of the method below, write the value that is returned:

```
def mystery(i, j):
    k = 0
    while (i > j):
        i = i - j
        k = k + (i - 1)
    return k
```

Function Call

Value Returned

mystery(2, 9)

mystery(5, 1)

mystery(38, 5)

mystery(5, 5)

mystery(40, 10)

2. Inheritance Mystery

Assume that the following classes have been defined:

```
class Pen(Sock):
    def method1(self):
        print("pen 1 ", end='')

class Lamp:
    def method1(self):
        print("lamp 1 ", end='')

    method2(self):
        print("lamp 2 ", end='')

    def __str__(self):
        return "lamp"
```

```
class Book(Sock):
    def method2(self):
        print("book 2 ", end='')
        super(Book, self).method2()

class Sock(Lamp):
    def method1(self):
        print("sock 1 ", end='')
        self.method2()

    def __str__(self):
        return "sock"
```

Given the classes above, what output is produced by the following code?

```
elements = [Book(), Pen(), Lamp(), Sock()]
for i in range(0, len(elements)):
    print(elements[i])
    elements[i].method1()
    print()
    elements[i].method2()
    print()
    print()
```

3. Collections Mystery

Consider the following method:

```
def mystery(list1, list2):
    result = new TreeMap<String, String>();
    for i in range(0, len(list1)):
        result[list1[i]] = list2[i]
        result[list2[i]] = list1[i]
    return result
```

The three entries below have specific values for the first and second parameters to method `mystery` (when you see text like the letter b, that indicates that the list stores the String "b" in that position). For each entry, indicate what values would be stored in the dictionary returned by function `mystery` if the given lists are passed as parameters. Dictionary elements should be listed with "key:value" elements, as in {b:z, d:e}.

list1: [b, l, u, e] list2: [s, p, o, t]

dictionary returned: _____

list1: [k, e, e, p] list2: [s, a, f, e]

dictionary returned: _____

list1: [s, o, b, e, r] list2: [b, o, o, k, s]

dictionary returned: _____

4. Collections Programming.

Write a function `union` that accepts two dictionaries (whose keys and values are both integers) as parameters, and returns a new dictionary that represents a merged union of the two original dictionaries. For example, if two dictionaries `m1` and `m2` contain these pairs:

```
{7=1, 18=5, 42=3, 76=10, 98=2, 234=50}      m1
{7=2, 11=9, 42=-12, 98=4, 234=0, 9999=3}    m2
```

The call of `union(m1, m2)` should return a dictionary that contains the following pairs:

```
{7=3, 11=9, 18=5, 42=-9, 76=10, 98=6, 234=50, 9999=3}
```

The "union" of two dictionaries `m1` and `m2` is a new dictionary that contains every key from `m1` and every key from `m2`. Each value stored in your "union" map should be the sum of the corresponding value(s) for that key in `m1` and `m2`, or if the key exists in only one of the two maps, that map's corresponding value should be used. For example, in the maps above, the key 98 exists in both maps, so the result contains the sum of its values from the two maps, $2 + 4 = 6$. The key 9999 exists in only one of the two maps, so its sole value of 3 is stored as its value in the result map.

Either dictionary passed in (or both) could be empty. Though the pairs are shown in sorted order by key above, you should not assume that the dictionaries passed to you store their keys in sorted order.

You may create one collection of your choice as auxiliary storage to solve this problem. You can have as many simple variables as you like. You should not modify the contents of the dictionaries passed to your function.

5. Programming

Write a static method called `encode` that takes a string `s` and an integer `n` as parameters and that returns a new string that scrambles the order of the characters from `s` in a particular way. Taking the characters from `s` in order, imagine filling a grid of `n` rows column by column. When `s` is "abcdefghijklmnopqrstuvwxy" and `n` is 3, you get:

```
row 1:  a d g j m p s v y
row 2:  b e h k n q t w z
row 3:  c f i l o r u x
```

The method should return the result of concatenating these characters together with row 1 first, then row 2, and then row 3. Notice that the final column will not necessarily be complete, as in the example above where the final column has only two characters. So if you make the call:

```
encode("abcdefghijklmnopqrstuvwxy", 3)
```

You should get back the string "adgjmpsvybehknqtwzcfilorux". The string might contain any characters, including spaces. For example, the call:

```
encode("four score and seven", 4)
```

returns "f rneosedvuc eroasn" because the following grid would be produced:

```
row 1:  f r n e
row 2:  o s e d v
row 3:  u c e
row 4:  r o a s n
```

You may assume that the string passed as a parameter is not empty and that the integer passed as a parameter is greater than or equal to 1 and less than the length of the string. You are not allowed to construct any structured objects other than strings to solve this problem (no list, list of lists, etc).

6. List of Lists Programming

Write a function called `num_unique` that takes a list of lists as a parameter and returns the number of unique values stored in it. For example, if you have the following list of lists:

```
lis = [[1, 2, 3], [4, 3, 2, 1], [6, 7, 7], [8]]
```

a call to `num_unique(lis)` should return 7. You may create one other data structure to help you solve this problem.

7. List Programming

Write a function named `longest_sorted_sequence` that accepts a list of integers as a parameter and that returns the length of the longest sorted (nondecreasing) sequence of integers in the list. For example, if a variable named `lis` stores the following values:

```
lis = [3, 8, 10, 1, 9, 14, -3, 0, 14, 207, 56, 98, 12]
```

then the call of `longest_sorted_sequence(lis)` should return 4 because the longest sorted sequence in the list has four values in it (the sequence -3, 0, 14, 207). Notice that sorted means nondecreasing, which means that the sequence could contain duplicates. For example, if the list stores the following values:

```
lis2 = [17, 42, 3, 5, 5, 5, 8, 2, 4, 6, 1, 19]
```

Then the method would return 5 for the length of the longest sequence (the sequence 3, 5, 5, 5, 8). Your method should return 0 if passed an empty list. Your method should return 1 if passed an list that is entirely in decreasing order or contains only one element.

8. Critters

Write a class `Bumblebee` that extends the `Critter` class from the Critters assignment.

A `Bumblebee` object should move in a "spiral" pattern from W to S to E to N, lengthening each time:

- one step west
- two steps south
- three steps east
- four steps north
- five steps west
- six steps south
- seven steps east
- eight steps north
- nine steps west
- ...

All other `Bumblebee` behavior uses the `Critter` defaults. You may add anything needed (fields, other methods) to implement this behavior appropriately.

9. Classes and Objects

Suppose that you are provided with a pre-written class `Rectangle` as described at right. (The headings are shown, but not the method bodies, to save space.) Assume that the fields, constructor, and methods shown are already implemented. You may refer to them or use them in solving this problem if necessary.

Write an instance method named `union` that will be placed inside the `Rectangle` class to become a part of each `Rectangle` object's behavior. The `union` method accepts another rectangle `r` as a parameter and turns this rectangle into the union of itself and `r`; that is, modifies the fields so that they represent the smallest rectangular region that completely contains both this rectangle and `r`.

For example, if the following `Rectangles` are declared in client code:

```
rect1 = Rectangle(5, 12, 4, 6)
rect2 = Rectangle(6, 8, 5, 7)

rect3 = Rectangle(14, 9, 3, 3)
rect4 = Rectangle(10, 3, 5, 8)
```

The following calls to the `union` method would modify the objects' state as indicated in the comments.

```
rect1.union(rect2)    # {(5, 8), 6x10}
rect4.union(rect3)   # {(10, 3), 7x9}
```

```
# A Rectangle stores an (x, y)
# coordinate of its top/left
# corner, a width and height.

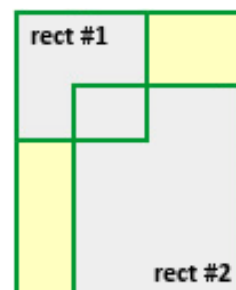
class Rectangle:

    # Constructs a new Rectangle
    # with the given x,y,w,h.
    def __init__(self, x, y, w, h)
        self.__x = x
        self.__y = y
        self.__width = w
        self.__height = h

    # returns the field values
    def getx(self)
    def gety(self)
    def get_width(self)
    def get_height(self)

    # example: {(5, 12), 4x8}
    def __str__(self)

    # your method would go here
```



CSc 110 Sample Midterm Exam #2 Solutions

1. While Loop Simulation

<u>Method Call</u>	<u>Value Returned</u>
mystery(2, 9)	0
mystery(5, 1)	6
mystery(38, 5)	119
mystery(5, 5)	0
mystery(40, 10)	57

2. Inheritance Mystery

```
sock
sock 1 book 2 lamp 2
book 2 lamp 2
```

```
sock
pen 1
lamp 2
```

```
lamp
lamp 1
lamp 2
```

```
sock
sock 1 lamp 2
lamp 2
```

3. Collections Mystery

```
list1: [b, l, u, e]
list2: [s, p, o, t]
dictionary returned: {b=s, e=t, l=p, o=u, p=l, s=b, t=e, u=o}
```

```
list1: [k, e, e, p]
list2: [s, a, f, e]
dictionary returned: {a=e, e=p, f=e, k=s, p=e, s=k}
```

```
list1: [s, o, b, e, r]
list2: [b, o, o, k, s]
dictionary returned: {b=o, e=k, k=e, o=b, r=s, s=r}
```

4. Collections

```
def union(m1, m2):
    result = {}
    for key, value in m1.items():
        result.put(key, value)
    for key, value in m2.items():
        if (key in result)
            result[key] = result.get(key) + value
        else:
            result[key] = value
    return result
```

5. Programming

```
def encode(s, n):
    result = ""
    for j in range(0, n):
        for i in range(0, len(s) - j, n):
            result += s[i + j]
    return result
```

6. List of Lists Programming

```
def num_unique(lis):
    unique = set()
    for i in range(0, len(lis)):
        for j in range(0, len(lis[i])):
            unique.add(lis[i][j])
    return len(unique)
```

7. List Programming

```
def longest_sorted_sequence(list):
    if (len(list) == 0):
        return 0

    max = 1
    count = 1
    for i in range(1, len(list)):
        if (list[i] >= list[i - 1]):
            count += 1
        else:
            count = 1

        if (count > max):
            max = count
    return max
```

8. Critters

```
class Bumblebee(Critter):
    public __init__(self) {
        self.__steps = 0
        self.__max = 1
        self.__direction = 0    # 0=west, 1=south, 2=east, 3=north

    def get_move(self):
        self.__steps += 1
        if (self.__steps > self.__max):
            # Pick a new direction and re-set the steps counter
            self.__steps = 1
            self.__max += 1
            self.__direction = (self.__direction + 1) % 4

        if (self.__direction == 0):
            return DIRECTION_WEST
        elif (self.__direction == 1):
            return DIRECTION_SOUTH
        elif (self.__direction == 2):
            return Direction.EAST;
        else: # self.__direction == 3
            return DIRECTION_NORTH
```

9. Classes

```
def union(r):
    # find the union's bounds
    left = min(self.__x, r.getx())
    top = min(self.__y, r.gety())
    right = max(self.__x + self.__width, r.getx() + r.get_width())
    bottom = max(self.__y + self.__height, r.gety() + r.get_height())

    self.__x = left
    self.__y = top
    self.__width = right - left
    self.__height = bottom - top
```