15-112
Fall 2021 Exam 1
September 28, 2021

Name:

Andrew ID:

- You may not use any books, notes, or electronic devices during this exam.
- You may not ask questions about the exam except for language clarifications.
- Show your work on the exam to receive credit.
- Write your answers in the specified places. If you run out of space for an answer, you may write on the backs of pages, but make sure to write a note telling the grader where to look for the rest of your answer.
- All code samples run without crashing. Assume any imports are already included as required.
- You may assume that math, string, and copy are imported; do not import any other modules.
- Do not use these post-midterm 1 topics/constructs: sets, maps/dictionaries, recursion, or classes/OOP.

Don't write anything in the table below.

Question	Points	Score
1	25	
2	20	
3	25	
4	15	
5	15	
6	0	
Total:	100	

1. **Code Tracing**

   (a) (10 points) Write the output for the following short code segments:

Code	**Output**

`x = 5+6*2**3-3`
`print(x)`	

`L = [16,8,32]`
`sorted(L)`
`print(L)`	

`L = [16,8,32]`
`L.sort()`
`print(L)`	

`a = [1,1,2]`
`b = [9,1,2,3,8]`
`for item in a:`
`    b[item] += 5`
`print(b)`	

`x = "Hello"`
`y = "Success"`
`while y[0] != "s":`
`    x = x + y[0]`
`    y = y[1:]`
`print(y,x)`	

`s = "ThereOnceWasAMan"`
`print(s.split("a"))`	

`a = True`
`b = False`
`print(a and (a or b) and not (a and b))`	

`def f():`
`    L = [2, 4, 6, 1]`
`    for item in L:`
`        if item % 2 == 1:`
`            return False`
`        else:`
`            return True`
`print(f())`	

`print(-7//5, -7%5)`	

`a = "112Rocks"`
`b = [2,3,4]`
`print(a[int(a[b[0]])+1:])`	

(b) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box next to the code.

```python
def ctb(n):
    r = 0
    while n > 0:
        print("n:",n)
        r *= 10
        r += n % 10
        n //= 100
    return r

print(ctb(9534875))
```

(c) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box next to the code.

```python
def g(a,n):
    while n>0:
        n //= 2
        a += [n]
    return sum(a)

def ctc(n):
    a= []
    b = 0
    while n > 0:
        d = n % 100
        if d % 3 == 0:
            b = b*100 + d
            print("left:",b)
        else:
            print("right:",d)
            print(g(a, d))
        n //= 10**(d%2 + 1)
        print(a, n)
n = 60105
ctc(n)
```

(d) (5 points) Indicate what the following program prints. Place your answers (and nothing else) in the box below the code.

```
def f(a,b,c):
    a += ["Bun"]
    b = b + ["Pup"]
    c = c[:] + ["Kit"]
    print("a3:",a)
    print("b3:",b)
    print("c3:",c)
    return c

def ctd(a):
    b = a
    c = b[:]
    b[1] = "Cat"
    c[0] = "Dog"
    print("a1:",a)
    print("b1:",b)
    print("c1:",c)
    b = f(c,b,a)
    print("a2:",a)
    print("b2:",b)
    print("c2:",c)

print(ctd(["GoGo","Success"]))
```

2. **Reasoning Over Code**

   (a) (10 points) Choose values for `x` and `y` to cause each of the following expressions to be `True`.

Code	x	y
`len(x) == 1 and x[0] == 112`		N/A
`x%y == 0 and x//y == 1`		
`x[::2].isupper() and x[1::2].islower()`		N/A
`x[0] == x[-1][0] and type(x) != type("42")`		N/A
`"level"[x:] == "level"[x::-1]`		N/A
`x == y.append(1)`		
`"15112"[x:] in "42"`		N/A
`y // x == x // y - 1`		
`sum(x) == x.count(y) and min(x) > 0`		
`int(x) // 10 % 10 == int(x[1]) and int(x) < 100`		N/A

(b) (5 points) Find the arguments for the following function to cause it to return True. Place your answer (and nothing else) in the box next to the code.

```
def rcb(n):
    assert(isinstance(n, int))
    assert(n > 99 and n < 1000)
    a = n % 10
    b = n // 10
    return a**2 == b and a + b == 72
```

(c) (5 points) Find the arguments for the following function to cause it to return True. Place your answer (and nothing else) in the box below the code.

```
def rcc(s,t):
    assert((s != "" and t != "") and len(s.split('-')) > len(t))
    n=0
    for i in s.split('-'):
        if len(i) != 1 or i.isdigit()==False:
            return False
        else:
            if int(i) % 3 == 0 and int(i) > 0:
                n += int(i)
                if s.count(i) != 1:
                    return False
            else:
                return False

    return n < 12 and sum(t) == n
```

3. **Free Response:** Stennes Numbers

   Do not use strings, lists, dictionaries, sets, try/except, or recursion on this problem. If you do, you will receive a 0.

   Well say that an integer is a stennes number (coined term) if it is a positive integer such that...

   - The digits are in descending order
   - The sum of all the digits is a multiple of 4
   - There are no 0s in the number
   - The number contains at least 4 digits.

   For example, 9852 is a stennes number because the digits are in descending order ($9 > 8 > 5 > 2$); the sum of the digits is 24, which is a multiple of 4; there are not 0s in the number; it contains 4 digits.

   For the purposes of defining descending order, a duplicated digit is not allowed. For example, 9852 is in descending order, but 9882 is not.

   The first 5 stennes numbers are: 5421, 6321, 6532, 6541, 7432.

   (a) (15 points) Write the function `isStennes(n)`, which takes a positive integer `n` and returns `True` if `n` is a stennes number and `False` otherwise. You can write any additional helper functions that you need.

(b) (10 points) Write the function `nthStennes(n)` which takes a non-negative integer `n` and returns the nth stennes number. `nthStennes(0)` should return `5421`, the first stennes number. You may assume that your implementation of `isStennes(n)` functions properly, even if yours does not.

4. (15 points) **Free Response:** Awkward Dates

The United States is one of the few countries that use *mm-dd-yyyy* as their date format – which is very very unique! In fact, in most countries the day is written first and the year last, for example *dd-mm-yyyy*.

Write the function `reformatDates(s)` which, given a string `s` that may contain American style (*mm-dd-yyyy*) dates, returns a copy of the string but with all of the valid dates changed to the international style (*dd-mm-yyyy*). For simplicity, you may assume that the days and months always include leading zeroes.

As an added complication, your code should only replace valid dates. For instance,

- **13-12-2021** is NOT a valid date in the *mm-dd-yyyy* format because there is no month 13.
- **09-31-2021** is NOT a valid date in the *mm-dd-yyyy* format because September has 30 days, and not 31 days.

Validating if a date is slightly complicated, and outside the scope of this question. To help you, we are providing the helper function `isValidDate(day, month, year)` that expects three `int` values: `day`, `month`, and `year`. It returns `True` if the date *day/month/year* is valid, and `False` otherwise. (You **do not** have to write this function. You can assume that it is there and use it in your code.)

Consider the following examples:

```
assert(reformatDates("Today is 09-28-2021")=="Today is 28-09-2021")
assert(reformatDates("Now 09-28-2021 14-34-59")=="Now 28-09-2021 14-34-59")
assert(reformatDates("Ups 13-00-2021")=="Ups 13-00-2021")
assert(reformatDates("12-01-2021 OK")=="01-12-2021 OK")
assert(reformatDates("Not a date 04-31-2021")=="Not a date 04-31-2021")
assert(reformatDates("Good luck")=="Good luck")
```

For your reference, here is the code for the provided helper function (but you really don't need to read it or care about it):

```
def isValidDate(day,month,year):
    # Number of days in the month
    daysInMonth = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    # Leap year check
    if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
        daysInMonth[1] += 1
    # Actually validate
    if month < 1 or month > 12 or day < 1 or day > daysInMonth[month-1]:
        return False
    return True
```

Answer space for Question 4

5. (15 points) **Free Response:** Word Ladders

A word ladder is a list of words (of the same length) that each differ from the previous word by exactly one letter. For example, the following is a word ladder:

`['hat', 'cat', 'bat', 'ban', 'pan']`

`cat` differs from `hat` at only the first letter, `bat` from `cat` at the first letter, `ban` from `bat` at the third letter, and `pan` from `ban` at the first letter.

Write the non-destructive function `isWordLadder(L)` which, given a list `L` of words, returns `True` if the words in `L` form a word ladder, and `False` otherwise. Consider the following examples:

```
assert(isWordLadder(['hat', 'cat', 'bat', 'ban', 'pan']) == True)
assert(isWordLadder(['cat', 'bat', 'tan', 'pan']) == False)
assert(isWordLadder(['cat', 'ban', 'bat']) == False)
assert(isWordLadder(['cat']) == True)
assert(isWordLadder([]) == True)
```

6. (2 points (bonus)) **Free Response:** Maybe Word Ladders

   This question is worth small points, and has no partial credit. Only work on it if you have extra time.

   Write the destructive function `wordLadderGame(L, newWord)` which, given a list `L` of words and a `newWord` returns `True` if the new word, inserted somewhere in `L`, produces a valid word ladder, and `False` otherwise. If the word can be part of a word ladder, modify `L` to include it. (If there are multiple correct places that `newWord` could be inserted, you may use any one of them.) You do not need to rearrange the existing elements of `L`. Instead, simply figure out if there is any place in `L` that you can insert `newWord` that produces a valid word ladder.