CMSC 201 Section 40
Spring 2022

Sample Final Exam

Note: this is taken from the final exam that I gave to the equivalent section in a previous semester. This is slightly different because we covered slightly different material this semester, but the idea is the same.

Section 1: Multiple Choice/ True-False – 16 questions, 4 points each

1. Which of the following is not a valid Python variable name?

   a. _time_of_day
   b. Time_of_day
   c. Time-of-day
   d. All of the above are valid

2. True or False: The “big-O” of an algorithm such as a sorting algorithm indicates how the algorithm will perform in the average case ***False***

3. True or False: If a list is unsorted, binary search is always guaranteed to either find a searched-for item or determine that it is not present in O(log n) time.***False***

4. If you want to go through a list, item by item, and change certain values in the list (e.g., change every floating point number into an integer), which of the following loops could you use?

a. for each loop

b. for I loop

c. while loop

d. both for I loops and while loops would work

5. If you wanted to open a file for writing, but you did not want to delete any information that might be there, in what mode would you open the file?

a. "a" or append mode

b. "w" or write mode

c. "r+" or reading and writing mode

d. this cannot be done in Python

6. In CMSC 201, class coding standards require that you write a function so that it only returns one variable. What would be the BEST WAY to handle a case where you really, really wanted to return multiple values from a single function?

a. Ignore the class coding standards; it's only a couple of points

b. combine the values into a list, and return the list

c. rewrite your code so you use multiple functions, each of which returns one of the values

d. write your program without using functions at all and eliminate the problem

7. Suppose I have the following definition:

```
def multiply_list(intlist, factor):
# this function multiplies each integer in intlist by factor and
returns a list of the products
newlist = intlist
for i  in range (len(newlist)):
      newlist[i] = factor * newlist[i]
return newlist
```

this function is called with the code

```
mylist = [1,2,3]
fact = 4
n = multiply_list(mylist, fact)
print(mylist)
```

What is the result of the print statement

a. An error; the statement fails to execute

b. [1,2,3]

c. [4,8, 12]

d. [1, 4, 9]

8. Now suppose I have the following definition:

```
def multiply_list(intlist, factor):
# this function multiplies each integer in intlist by factor and
returns a list of the products
newlist = []
for i  in range (len(intlist)):
     newlist.append( factor * intlist[i])
return newlist
```

this function is called with the code

```
mylist = [1,2,3]
 fact = 4
  n = multiply_list(mylist, fact)
 print(mylist)
```

What is the result of the print statement

a. An error; the statement fails to execute

b. [1,2,3]

c. [4,8, 12]

d. [1, 4, 9]

9. True or False: you can create a dictionary by zipping together two lists of the same length.***True***

10. What types can be used as values in a Python dictionary?

a. only types like string, int and float

b. only types like lists and dictionaries

c. any type at all can be a value

d. none of the other answers is correct.

11. Which of the following is a difference between lists and dictionaries in Python?

a. lists are ordered; dictionaries are unordered

b. Lists can be multi-dimensional; that is, be lists of lists; but dictionaries cannot contain dictionaries as values

c. you can have a list whose elements are lists, but you can't have a dictionary whose values are dictionaries

d. None of the above is correct

12. A file called numbers.py contains a python program I wrote last year. If I import numbers.py into my current python program, I can use code in the main program of numbers.py in my current program. ***False***

13. A valid recursive program can have how many base cases?

a. 0 or 1

b. exactly 1

c. at least 1

d. any number

14. Which is the correct way to instantiate a new, empty 2D list in Python?

a. l = [ ] []

b. l = [ ]

c. l = {}

d. ; = {[ ]}

15. If I import the Python math library into my program using the statement

```
from math import *
```

What is the correct way to call the arc tangent routine, atan?

a. `Y = math.atan(x)`

b. `Y = atan(x)`

c. `Y = math(atan(x))`

d. None of the other answers is correct

16. Suppose I have the following Python code:

```
j = 9/2
k = 9//2
if j == k:
      print("yes")
else:
      print("no")
```

True or false: the result of this code is that "no" is printed. ***True***

Section 2: Short answer – 13 questions; 7 points each

17. Why is quicksort considered to be faster than bubble sort, even though in the worst case they're both O(n**2)?

Because quicksort is much faster in the average case. It's only the same in the worst case, and the worst case rarely happens in real life.

18. When you are reading from a file, what does the file pointer do?

Tells what the next character to be read is

19. Why is a sentinel while loop a good structure to use if you are getting input from a user at the keyboard? Think about whether the user will enter the value you want the first time he sees the prompt.

A sentinel loop reads until you see a specific desired value. If the user enters that the first time - e.g., the user's first input is "q" to quit - you never have to execute the loop.

20. What can happen in a recursive program if the recursive case in a function does not involve a simpler version of the original problem?

An infinite recursion

21. Suppose that s is a string with the value "Nebraska" Why is permissible to include a Python statement like

```
      if s[0]== "N":
```

But not a statement like

```
s[1] = "A"
```

Strings are immutable. You cannot change them in place. You can read them a character at a time, because reading doesn't change them.

22. Suppose you had a function, factorial, defined as follows:

```
def factorial (number):
   if number < 0:
      return -1
  elif number == 0:
      return 1
  else:
      return number * factorial (number -1)
```

And your main program was:

```
if __name__ == "__main__":
     n = 5
     k = factorial(n)
     print(k, "is the factorial of ", number)
```

The result is that you get the error message "number is not defined". Why?

"Number" is a local variable in the function. You're trying to use it in the main program, where it is not defined.

23. What characters are allowed to start a Python variable name?

Underscores, upper and lower case letters

24. This semester we learned about a number of sorting algorithms including bubble sort and selection sort. The worst case performance of both is O(n-squared), but the best case performance of bubble sort is better – Omega (n) vs. Omega (n-squared). Explain why.

Bubble sort allows you to "short-circuit" the loop and stop executing if you have gone through the list without swapping values. If you go through without swapping, that means the list is already sorted. Selection sort does not allow this shortcut.

25. Suppose that states is a one-dimensional list of state names in alphabetical order, and senators is a two-dimensional list of the 100 US senators. It has 50 rows; each row contains the names of the two Senators from that state. Write the Python code that lets you create a dictionary, the_senate, from these two lists. The keys are the state names; the values are the names of the two Senators from that state. You can presume that the two lists are in the proper order; e.g., the first row of senate contains the two Senators from the first state in states.

Answer = dict(zip(states,senators))

26. Write a Python function that reads in a tab-separated value (.tsv) file, t.tsv, as a string and splits it into a two-dimensional list. You may presume that .tsv files, like .csv files, have each line of data end in a newline character, "\n"

```
def split(tabfile):
 with open (tabfile, "r") as infile:
        data = infile.readlines()
         for i in len(data);
                data[i] = data[i].split("\t")
return data
```

27. Suppose you have a list of the states in random order, like:

```
states =

['Idaho', 'Utah', 'Hawaii', 'Maine', 'New York', 'South Carolina',
'Illinois', 'Pennsylvania', 'North Carolina', 'Colorado',
'California', 'Alaska', 'Missouri', 'Kansas', 'Oklahoma',
'Connecticut', 'South Dakota', 'North Dakota', 'Louisiana', 'Nevada',
'Delaware', 'Washington', 'Wisconsin', 'Georgia', 'Nebraska',
'Virginia', 'Wyoming', 'New Hampshire', 'Texas', 'Kentucky', 'West
Virginia', 'Rhode Island', 'Maryland', 'Massachusetts', 'Vermont',
'New Mexico', 'Florida', 'Tennessee', 'Iowa', 'Arizona', 'Montana',
'Minnesota', 'Alabama', 'Mississippi', 'Arkansas', 'Oregon', 'New
Jersey', 'Ohio', 'Michigan', 'Indiana']
```

Why won't the binary search algorithm be a good way to find the location of "Maine"?

Binary search only works for sorted files. If you tried it here, you would jump to the middle value, which is "Nebraska." "Maine" is less than "Nebraska," so you'd take the first half of the file. You'd jump to the middle value, which is "Missouri." You'd take the first half of that list. Jump to the middle value, which is "Illinois." "Maine" is greater than "Illinois" so you'd take the last half of that list, but that's not where "Maine" is. You'd never find "Maine"; you'd report that it wasn't there.

28. Why do we as computer scientists care so much about sorting algorithms, when sorting is usually done much less often than operations like searching?

Because things like binary search only work with sorted lists. So it's worthwhile to sort your list, and get the huge advantage of O(log n) binary search over O(n) linear search.

29. Explain why you might write a program recursively, even though you know you could always solve it iteratively with while loops instead.

Some programs are naturally easier to understand when written recursively.

Section 3: Programming – 3 questions, 15 points each

30. Write a Python program that asks the user to enter the user's semester grade in CMSC 201 as an integer between 0 and 1000, inclusive. If the user enters a number between 900 and 1000, inclusive, print out the message "congratulations on your A." If the user enters a number between 800 and 899, inclusive, print out the message "a B is not a bad grade." If the user enters a number above 1000 or below 800, print out the message "I don't believe that's a valid answer" and ask for the grade again. Continue doing this until you get a valid grade.

If the user enters something that is not an integer, it is okay if your program crashes.

```
score = int(input("Enter the student's grade"))

while score > 1000 or score < 800:

   print("I don't believe that's a valid grade")

   score = int(input("Enter the student's grade"))

if score > 900:

      print("congratulations on your A")

elif score > 800:

   print("a B is not a bad grade")
```

31. Write a recursive Python program that reads a string from the last character to the first and prints out the message “found an x!” every time it encounters the letter “x” in the string. You MUST use a recursive function in your solution!

Base case: string is 1 character or 0 characters long

Recursive case: delete last letter & call function with rest of string - string slicing - e.g., s[:-1]

```
def check_for_x(s):
      if len(s) == 0:
            return None
      else:
            if s[-1]== “x”:
                  print(“Found an X!”)
            check_for_x(s[:-1]


if __name__ == “__main__”:
      str = input(“Enter the string to check”)
      check_for_x(str)
```

32. Insertion sort is an algorithm that sorts a list by first creating a new, empty list. Then it takes the elements of the unsorted list, one at a time, and inserts them in the proper place in the new list. That is: for each element in the original list, start at the beginning of the new list and find the first element that is greater than this new element, and insert the new element immediately before that. If there are no elements greater than this new element, append it to the end of the new list.

Write an interactive function in Python that implements insertion sort. You do not need to write the entire program; just the function. It should take one parameter, which is a list; and return a new, sorted list.

```
 def insertion_sort(unsorted_list):
      sorted_list = [ ]
```

```
for i in range(len(unsorted_list)):
      #put the element in the right place
      j = 0
      found = False
      while j < len(sorted_list)) and not found:
            if sorted_list[j] > unsorted_List[i]:
                  sorted_list.insert(j,unsorted_list[i])
                  found = True
            j += 1
return (sorted_list)
```