# CMPT 120 D300 Final

Veronica Diaz Lopez

TOTAL POINTS

**195 / 252**

QUESTION 1

*1* Q1 **7 / 10**

- **0 pts** Correct
- **1 pts** One incorrect answer
- **2 pts** Two incorrect answers
- ✓ **3 pts** *Three incorrect answers*
- **4 pts** Four incorrect answers
- **5 pts** Five incorrect answers
- **6 pts** Six incorrect answers
- **7 pts** Seven incorrect answers
- **8 pts** Eight incorrect answers
- **9 pts** Nine incorrect answers
- **10 pts** Ten incorrect answers

QUESTION 2

*2* Q2 **16 / 20**

- **0 pts** Correct
- **2 pts** One incorrect answer
- ✓ **4 pts** *Two incorrect answers*
- **6 pts** Three incorrect answers
- **8 pts** Four incorrect answers
- **10 pts** Five incorrect answers
- **12 pts** Six incorrect answers
- **14 pts** Seven incorrect answers
- **16 pts** Eight incorrect answers
- **18 pts** Nine incorrect answers
- **20 pts** Ten incorrect answers

QUESTION 3

Q3 40 pts

*3.1* Q3.1 **0 / 4**

- **0 pts** Correct
- ✓ **4 pts** *Incorrect*

*3.2* Q3.2 **4 / 4**

- ✓ **0 pts** *Correct*
- **4 pts** Incorrect

*3.3* Q3.3 **4 / 4**

- ✓ **0 pts** *Correct*
- **4 pts** Incorrect

*3.4* Q3.4 **4 / 4**

- ✓ **0 pts** *Correct*
- **4 pts** Incorrect

*3.5* Q3.5 **0 / 4**

- **0 pts** Correct
- ✓ **4 pts** *Incorrect*

*3.6* Q3.6 **4 / 4**

- ✓ **0 pts** *Correct*
- **4 pts** Incorrect

*3.7* Q3.7 **0 / 4**

- **0 pts** Correct

✓ **- 4 pts** *Incorrect*

*3.8* Q3.8 **0 / 4**

**- 0 pts** Correct

✓ **- 4 pts** *Incorrect*

*3.9* Q3.9 **0 / 4**

**- 0 pts** Correct

✓ **- 4 pts** *Incorrect*

*3.10* Q3.10 **0 / 4**

**- 0 pts** Correct

✓ **- 4 pts** *Incorrect*

QUESTION 4

## Q4 180 pts

*4.1* Q4.1 **18 / 18**

✓ **- 0 pts** *Correct*

**- 1.5 pts** Minor syntax error #1

**- 1.5 pts** Minor syntax error #2

**- 1.5 pts** Minor syntax error #3

**- 2.5 pts** Major syntax error #1

**- 2.5 pts** Major syntax error #2

**- 2.5 pts** Minor semantic error

**- 5 pts** Modertate semantic error

**- 7 pts** Major semantic error

*4.2* Q4.2 **11 / 18**

**- 0 pts** Correct

**- 1.5 pts** Minor syntax error #1

**- 1.5 pts** Minor syntax error #2

**- 1.5 pts** Minor syntax error #3

**- 2.5 pts** Major syntax error #1

**- 2.5 pts** Major syntax error #2

**- 2.5 pts** Minor semantic error

**- 5 pts** Modertate semantic error

✓ **- 7 pts** *Major semantic error*

*4.3* Q4.3 **18 / 18**

✓ **- 0 pts** *Correct*

**- 1.5 pts** Minor syntax error #1

**- 1.5 pts** Minor syntax error #2

**- 1.5 pts** Minor syntax error #3

**- 2.5 pts** Major syntax error #1

**- 2.5 pts** Major syntax error #2

**- 2.5 pts** Minor semantic error

**- 5 pts** Modertate semantic error

**- 7 pts** Major semantic error

*4.4* Q4.4 **18 / 18**

✓ **- 0 pts** *Correct*

**- 1.5 pts** Minor syntax error #1

**- 1.5 pts** Minor syntax error #2

**- 1.5 pts** Minor syntax error #3

**- 2.5 pts** Major syntax error #2

**- 2.5 pts** Major syntax error #1

**- 2.5 pts** Minor semantic error

**- 5 pts** Modertate semantic error

**- 7 pts** Major semantic error

*4.5* Q4.5 **18 / 18**

✓ **- 0 pts** *Correct*

**- 1.5 pts** Minor syntax error #1

**- 1.5 pts** Minor syntax error #2

**- 1.5 pts** Minor syntax error #3

**- 2.5 pts** Major syntax error #1

**- 2.5 pts** Major syntax error #2

**- 2.5 pts** Minor semantic error

- **5 pts** Modertate semantic error
- **7 pts** Major semantic error

### *4.6* Q4.6 **18 / 18**

✓ **- 0 pts** *Correct*
- **1.5 pts** Minor syntax error #1
- **1.5 pts** Minor syntax error #2
- **1.5 pts** Minor syntax error #3
- **2.5 pts** Major syntax error #1
- **2.5 pts** Major syntax error #2
- **2.5 pts** Minor semantic error
- **5 pts** Modertate semantic error
- **7 pts** Major semantic error
- **18 pts** Not answered

### *4.7* Q4.7 **6.5 / 18**

- **0 pts** Correct
- **1.5 pts** Minor syntax error #1
- **1.5 pts** Minor syntax error #2

✓ **- 1.5 pts** *Minor syntax error #3*

✓ **- 2.5 pts** *Major syntax error #2*

✓ **- 2.5 pts** *Major syntax error #1*
- **2.5 pts** Minor semantic error

✓ **- 5 pts** *Modertate semantic error*
- **7 pts** Major semantic error

### *4.8* Q4.8 **15.5 / 18**

- **0 pts** Correct
- **1.5 pts** Minor syntax error #1
- **1.5 pts** Minor syntax error #2
- **1.5 pts** Minor syntax error #3
- **2.5 pts** Major syntax error #1
- **2.5 pts** Major syntax error #2

✓ **- 2.5 pts** *Minor semantic error*
- **5 pts** Modertate semantic error
- **7 pts** Major semantic error
- **18 pts** No answer

### *4.9* Q4.9 **13 / 18**

- **0 pts** Correct
- **1.5 pts** Minor syntax error #1
- **1.5 pts** Minor syntax error #2
- **1.5 pts** Minor syntax error #3
- **2.5 pts** Major syntax error #1
- **2.5 pts** Major syntax error #2
- **2.5 pts** Minor semantic error

✓ **- 5 pts** *Modertate semantic error*
- **7 pts** Major semantic error

### *4.10* Q4.10 **18 / 18**

✓ **- 0 pts** *Correct*
- **1.5 pts** Minor syntax error #1
- **1.5 pts** Minor syntax error #2
- **1.5 pts** Minor syntax error #3
- **2.5 pts** Major syntax error #1
- **2.5 pts** Major syntax error #2
- **2.5 pts** Minor semantic error
- **5 pts** Modertate semantic error
- **7 pts** Major semantic error
- **18 pts** No Answer

## QUESTION 5

### *5* Q5 **2 / 2**

✓ **- 0 pts** *Awsome!*

000053 0001

SFU

SIMON FRASER UNIVERSITY

## School of Computing Science

Final Exam, Spring 2023

# CMPT 120 Introduction to Computing Science and Programming

Exam Duration: 150 minutes

Name Veronica Diaz Lopez

Student Number 301575755

**Exam Conditions:**

- Print your name and student number clearly above.
- This is a paper-based and a closed-book examination.
- The exam duration is 150 minutes.
- No electronic aids are permitted e.g. laptops, or phones.
- One blank scrap paper is permitted.
- Questions in this exam implicitly refer to the Python programming language.
- You must record all your answers in the spaces provided in this document.
- Use the white space on page 16 to write an answer only if needed and by adding a note in the original place.

| Question | Mark |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 3 | 40 |
| 4 | 180 |
| 5 | 2 |
| Total | 250 + 2 |
| | |

**Part 1 (10 points)**
**For each of the following statements, indicate whether it is TRUE or FALSE.**

1.1) Variable names can be arbitrarily long.

1.2) ASCII is a way of representing numbers using binary.

1.3) Lists are mutable, but strings are immutable.

1.4) A recursive function will run forever if it does not include a base case.

1.5) Bubble sort uses more memory than merge sort to sort a list of numbers.

1.6) In a dictionary, items can be accessed by their position in a dictionary.

1.7) The concatenation operator (+) concatenates two lists by appending one list at the beginning of the other list.

1.8) Function definitions do not alter the flow of execution of the program.

1.9) Extending a list with a string will append all the characters of the string at the end of the list.

1.10) O(n!) shows a lower time complexity compared to O(n ^ 3).

Part 1 - Answers

1.1 True ✓
1.2 False ✓
1.3 False ✗ ✓
1.4 True ✓
1.5 True ✗ ✓
1.6 False ✓
1.7 False ✓
1.8 False ✗
1.9 True ✓
1.10 False ✓

**Part 2 (20 points)**
**Select the correct answer.**

2.1) Which statement produces an error? Assume variables a and b have already been defined.

a) a * b + b // 4 –
b) a = a + 2b * 2
c) a = a + a, a, b - 3 -
d) a = a ** a ** b ** a

a a b^a

2.2) What is the output of the following code segment?

a) 4 2
b) 2 2
c) 2 4
d) 4 4

```
x = (3 % 4) + 1
y = (4 % 3) + 1
print(x, y)
```

16
7 + 9
2 + 5 + 9
2 + 12 - 7 + 9

2.3) What is the output of the following code segment?

a) 22
b) 16
c) 13
d) 9

```
a = 2 + (3 * 4) - 7 + (3 ** 2)
print(a)
```

2.4) The following code is executed, and at the prompt the user enters **3**. What is the output of this program?

a) 6
b) 4
c) This program produces an error.
d) None of above

```
def get_input():
    x = int(input("Enter a number: "))
    return x
def main():
    print(get_input() * 2)
main()
```

3 3 * 2

2.5) What is the output of the following code segment?

a) book
b) koob
c) b o o k
d) k o o b

```
s = ['b', 'o', 'o', 'k']
t = ""
for i in range(len(s)):
    t = s[i] + t
print(t)
```

bob oob Koob

2.6) What is the output of the following code segment?

a) This program produces an error.
b) -2101
c) 2102
d) -1012
e) 1012

```
s = ''
for i in range(-1, 3):
    s = s + str(i)
x = int(s)
print(x)
```

-1012

2.7) What is the time complexity of the following program?

a) O(n)
b) O(n^2)
c) O(n^3)
d) O(n^4)

```
for i in range(n):
    for j in range(1, n):
        print(i, j)
    for p in range(1, n):
        p = p * i
for k in range(1, n):
    k = k + 1
```

2.8) What is the output of the following code segment?

a) 2
b) 4
c) 6
d) 8

```
source = '10'
counter = 10
while counter < 30:
    source += source
    counter *= 6
print(len(source))
```

counter = counter * 6
10 → 60

1010

2.9) What is the output of the following code segment?

a) 10
b) 20 3040
c) 10 20 3040
d) 1020 3040

```
f = open("test.txt", "w")
f.write("10")
f.close()
f = open("test.txt", "a")
f.write("20 3040")
f.close()
f = open("test.txt", "r")
print(f.read())
```

2.10) What is the output of the following code segment?

when p = 1
1 → 2
2 → 4
4 → 8

a) 2
b) 4
c) 6
d) 8

```
def rec_func(n, p):
    if p == 1:
        return 1
    else:
        return n * rec_func(n, p - 1)
print(rec_func(2, 3))
```

**Part 2 - Answers**

2.1 a ✓
2.2 b ✓
2.3 b ✓
2.4 a ✓
2.5 b ✓
2.6 d ✓
2.7 d ✗
2.8 b ✓
2.9 d ✓
2.10 d ✗

**Part 3 (40 points)**
**What is the output of the following code segment?**

**Question 3.1**

```
print("Life is","beautiful", end="!")
```

**Answer 3.1**

Life isbeautiful!

**Question 3.2**

```
values = [-4, 12, 100, 5, 12, 6, 120, 5]
result = [1]
for value in values:
    if 5 <= value < 12:
        result.append(value)
print(result)
```

**Answer 3.2**

[1, 5, 6, 5]

**Question 3.3**

```
x = 3
while x > 0:
    print(x + 1)
    x = x - 1
```

**Answer 3.3**

4
3
2

**Question 3.4**

```
y = 1
x = (2 * 4) - 8
print (x != y)
```

x - 8 = 0 x != 4 True

**Answer 3.4**

True

**Question 3.5**

```
d = {'z': 10, 'b':1, 'c' :29}

for x in sorted(d.values()):

    print(x, end=" ")
```

**Answer 3.5**

10 1 29

**Question 3.6**

```
def rec_func(n):
    if n < 3:
        print(n, "Start")
        rec_func(n + 1)
        print(n, "End")
    else:
        print(n, "Start")
        print(n, "End")

rec_func(2)
```

**Answer 3.6**

2 Start
3 Start
3 End
2 End

**Question 3.7**

```
lst = ['b', ['cc', 'dd', ['eee', 'fff']], 'g', 'h']
print(lst[1][2][1] + lst[3])
```

**Answer 3.7**

['eee', 'h']

E > M
D > M

**Question 3.8**

```
lst = [x*3 for x in 'RED' if x > 'M']

print(lst)
```

**Answer 3.8**

[EEE, DDD]

**Question 3.9**

```
my_dict = {}

for i in 'book':

    for j in range(3):

        my_dict[i] = j

print(my_dict)
```

**Answer 3.9**

{b = 0, o = 1, o = 2} AMA

**Question 3.10**

```
x = 'red'
          321
while len(x) > 0:

    print(x[:-1])

    x = x[1:]
```

**Answer 3.10**

re
e
d

**Part 4 (180 points)**

Question 4.1) Write a program to take three numbers from the user and print the largest one.

Answer 4.1

```
#input
n1 = int(input(""))
n2 = int(input(""))
n3 = int(input(""))
#numbers in list
num_list = [n1, n2, n3]
# print max.
print(max(num_list))
```

Question 4.2) Write a function to take a string and an integer **n** and print all substring of size **n** of the given string. Note that a substring is a contiguous sequence of characters within a string.

Answer 4.2

```
#input
string = input("")
n = int(input(""))
def substring(string, n): new_str = ""
    for j in range(len(string)):
        for i in range(n+1):
            new_str += string[i]
        return(new_str)
print(substring(string,n))
```

Question 4.3) Write a function to receive a list of integers and compute the mean of these values. Mean refers to the average of a set of values.

Answer 4.3

```
num_list = [# input received from user].
def avg(num_list):
    total = 0
    count = 0
    for num in num_list:
        total += num
        count += 1
    return total // count
print(avg(num_list))
```

Question 4.4) Write a function to take a string **source** and a character ***k***, and return the length of the longest substring in **source** containing only character **k**. E.g. the function would return 4 given ('booookko', 'o'), and 3 given ('aaabbcbbccaa', 'a'). Note that a substring is a contiguous sequence of characters within a string.

Answer 4.4

```
def sub_string(source, k):
    for i in source:
        if i == k:
            count += 1
        else
            sub.append(count)
            count = 0
    return max(sub)
```

```
# look for k in source
# append to list if next to.
# increase count by 1.
# print count.
```

```
def sub_string(source, k):
    sub = [ ]
    count = 0
    for i in source:
        if i == k:
            count += 1
        else:
            sub.append(count)
            count = 0
    return max(sub)
```

Question 4.5) Write a function to take two dictionaries each with integer keys and integer values and return a list including keys that are *even* and common in these two dictionaries. E.g. this function should return [2] if it receives {1:4, 2:5, 3:9} and {1:2, 2:7, 9:3}.

Answer 4.5

```
def even_key(dict1, dict2):
    key_list1 = []
    key_list2 = []
    for key in dict1.key():
        key_list1.append(key)
    for key in dict2.key():
        key_list2.append(key)
    if num in key_list1 and key_list2:
        if num % 2 == 0:
            even_key.append(num)
return even_key
```

even_key = []

Question 4.6) Write a program to read a file called 'words.txt' and search for a word obtained from the user and print the number of times the given word appeared in the file. Words in the file are separated by space. Your program should be *case-insensitive*, meaning that it should not discriminate between uppercase and lowercase letters.

Answer 4.6

```
file = open("words.txt", "r")
words = file.readlines()
for word in words:
    word_list.append(lowercase(word))
word_list = []
word_list.appen
for lines in file.read().split(" ")
    word_list.append(lines)
dict = { }
for word in word_list
    dict[word] = 1
    if word in dict[key]
        dict[word] += 1
    else
        dict[word] = 1
```

space

```
if given_word
in dict.key()
print dict(given_word).
```

# read line in file
# split words by space
# add word to dictionary and value is count 1
# if word already in dict increase value by 1.
look for given word in dictionary key and print value

Question 4.7) Assume a dictionary with keys as the student's name and values a list of numbers (between 0 and 100) as the student's grades. Write a function to take such a dictionary and a file name and write students' information in that file. Each line of file should include three pieces of information separated by space: the student's name, the minimum grade of the student, and the maximum grade of the student. You are allowed to use Python's built-in *min()* and *max()* functions in your program.
For example, for parameters ({'Emil' : [85, 93], 'Emily' : [92, 80, 94]}, 'grades.txt') the content of the file 'grades.txt' should be as follows after calling the function:

Emil 85 93
Emily 80 94

Answer 4.7

```
def grades(dict, file)
    for key in dict.keys():
        min_max = [ ]
        min_max.append(min(dict.get()))
        min_max.append(max(dict.get()))
        file = open("grades.txt", "w")
        file.write(key, int(min_max[0], int(min_max[1])
    file.close()
```

dict(student) = [grades]
file = grades.txt

Question 4.8) Write a function to take numbers **a** and **b** and return the count of the **odd** numbers between **a** and **b** (inclusive) that are also a multiple of **5**. For example, for parameters (2, 23) this function returns 2. Note that **a** is not necessarily smaller than **b**.

**Answer 4.8**

```
def find_num(a, b)
    count = 0 → odd_list = []
    for num in range(a, b+1):
        if num % 2 == 0:
            count = count
        else:
            odd_list.append(num)
    for num in odd_list:
        if num % 5 == 0:
            count += 1
        else:
            count = count
    return count
```

Question 4.9) Consider the following list:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [ | 'a' , | 'c' , | 'd' , | 'e' , | 'h' , | 'i' , | 'j' , | 'm' , | 'p' , | 'q' , | 'r' , | 's' , | 'v' , | 'x' , | 'z' ] |
| (index) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |

Show how a binary search would proceed in finding the character 'x' in this list by **only** showing the values of **L**, **R**, and **M** at each step of the search process.

~~left → right~~ middle.

**Answer 4.9**

Iteration #1.
Looking for x → index 13
L = 0 R = 14 M = 7 M > 13 > R
discard index 0 to 6.

Iteration #2.
Looking for x → 13 L = 7 M = 10 R = 14
M > 13 > R
discard index 7 to 9

Iteration #3 Looking for x → 13
L = 10 M = 12 R = 14
M > 13 > R
discard index 10 to 11

Iteration #4
Looking for x → 13
L = 12 M = 13 R = 14
X = M = 13 ✓

Question 4.10) Given a list of numbers, **lst**, and a number, **x**, write a *recursive* function to count how many of **x** are in **lst**. For example, this function returns 3 for parameters ([1,2,8,2,2], 2).

Answer 4.10

```
def count_num(lst, x):
    count = 0
    if len(lst) == 0:
        if lst[0] == x
            return 1
    else:
        return count + count_num(lst[1:], x):
```

1 [1,8,2,2], 2
2 [8,2,2], 2 c = 2
3 [2,2], 2
4 [2], 2

**Part 5 (2 points)**
Bonus question: Draw a hacker face!

Answer 5

**END OF EXAMINATION**

Extra White Space