CMPT 120 Standard Final Exam

Sample 1
Multiple Choice Questions

Duration	1 hour
Aids allowed	No notes, no papers, no books, no computers, no calculators, etc.
Scoring	For each question fill in **the one best answer** on the answer sheet. Correct answers are worth 1 point. Incorrect answers, multiple answers, illegible answers, or unanswered questions are worth 0 points.
During the exam	Raise your hand if you have a question and remain seated. We will come to you. Questions about exam/course content will **not** be answered during the exam.

1) What does this print?

```
print('3' + '4' * 2)
```

A. 11
B. 14
C. 344
D. 3434
E. nothing is printed: the statement has an error

2) Consider this code fragment:

```
a = 2.5
b = 3.2
a = b + a
b = a + b
print(a)
print(b)
```

i) The printed value of `a` is 5.7
ii) The printed value of `b` is 8.9

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

3) Consider this code fragment:

```
a = 2
b = 7
???
print(a) # 7
print(b) # 2
```

Suppose **???** is replaced by one of the fragments below. Which one makes the code print 7 for `a` and 2 for `b`?

A.
```
  a = a - b
  b = b + a
  a = a - b
```
B.
```
  t = a
  a = b
  b = t
```
C.
```
  t = b
  a = b
  b = t
```
D. all of A, B, and C
E. none of A, B, or C

4) Consider this statement:

```
print(2 ??? (10 % 4))  # 4
```

How many of these 4 operators can replace **???** so that the statement prints 4?

```
+   *   **   %
```

A. 0 of the operators
B. 1 of the operators
C. 2 of the operators
D. 3 of the operators
E. 4 of the operators

5) Consider these statements:

i) Python strings can be modified
ii) Python lists can be modified

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

6) What does this print?

```
lst = [2, 0, -1, 1]
print(lst[lst[1]])
```

A. 2
B. 0
C. -1
D. 1
E. nothing is printed: there is an indexing error

7) What does this print?

```
S = [2, 1, 3]
T = S
S[2] = 0
print(T)
```

A. `[2, 1, 3]`
B. `[2, 1, 0]`
C. `[2, 0, 3]`
D. `[0, 1, 3]`

8) Consider this code:

```
s = 'lizard'
```

Which statement prints `zar` ?

A. `print(s[2:4])`
B. `print(s[2:5])`
C. `print(s[3:5])`
D. `print(s[3:6])`

9) Consider these statements:

i) A dictionary can have duplicate keys.
ii) A dictionary can have duplicate values.

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

10) How many of these three programs print 6?

```
# program 1
d = {'a':1, 'b':2, 'c':3}
total = 0
for x in d:
    total += x
print(total)
```

```
# program 2
d = {'a':1, 'b':2, 'c':3}
total = 0
for x in d:
    total += d[x]
print(total)
```

```
# program 3
d = {'a':1, 'b':2, 'c':3}
total = 0
for x in range(len(d)):
    total += d[x]
print(total)
```

A. 0
B. 1
C. 2
D. 3

11) What does this program print if the user enters `house` for `a` and `mouse` for `b`?

```
a = input('a? ')
b = input('b? ')
if not (len(a) != len(b)):
    print('yes')
if len(a) == len(b):
    print('no')
print('done')
```

A.
```
done
```
B.
```
yes
done
```
C.
```
no
done
```
D.
```
yes
no
done
```

12) Assuming both `a` and `b` are strings, which code fragment prints `'good'` exactly when `a` and `b` are different strings of the same length?

A.
```
if len(a) == len(b) and a != b:
    print('good')
else:
    print('bad')
```
B.
```
if len(a) == len(b) or a != b:
    print('good')
else:
    print('bad')
```
C.
```
if not (len(a) == len(b) and a == b):
    print('good')
else:
    print('bad')
```
D.
```
if not (len(a) == len(b) or a == b):
    print('good')
else:
    print('bad')
```

13) What values of `a` and `b` make this code print 2?

```
if a < 0 or b < 0:
    print(a)
elif a < b < 0:
    print(b)
else:
    print(a + b)
```

A. `a` is 2, `b` is 2
B. `a` is 2, `b` is -1
C. `a` is -1, `b` is 2
D. `a` is -1, `b` is -1
E. none of the above values of `a` and `b` that make the code print 2

14) What function call returns the same value as `f('4')` ?

```
def f(c):
    result = 0
    if c in '0123456789':
        if c in '01':
            result += int(c)
        if c in '02468':
            result += int(c)
        else:
            result += int(c) - 1
    else:
        result = -1

    return result
```

A. `f('2')`
B. `f('3')`
C. `f('5')`
D. `f('6')`
E. none of the above return the same value as `f('4')`

15) If variables `a` and `b` are both strings, what are the possible values of this expression?

```
    (a == b) and (a != b)
```

A. it always evaluates to `True`
B. it always evaluates to `False`
C. depending upon the values of `a` and `b`, sometimes it evaluates to `True`, and sometimes it evaluates to `False`

16) What does this print?

```
x = 2
result = 0
for i in range(5):
    if i > 2:
       result += i
print(result)
```

A. 7
B. 10
C. 12
D. 15

17) What does this print?

```
s = 'grads'
result = ''
for i in s:
    if i < 'k':
        result += i
print(result)
```

A. nothing is printed because the final value of `result` is the empty string
B. `rs`
C. `gad`
D. the program crashes when `i < 'k'` is evaluated

18) What does this print?

```
lst = [4, 0, 9, 1]
result = 0
for i in range(len(lst)):
    result += i + lst[i]
print(result)
```

A. 6
B. 14
C. 20
D. an int other than 6, 14, or 20

19) What does this print?

```
result = 'start'
for i in ['up', 'moose', 'elephant', '!']:
    if len(i) > len(result):
        result = i
print(result)
```

A. `start`
B. `up`
C. `moose`
D. `elephant`
E. `!`

20) What does this print?

```
result = 0
for i in range(5):
    for j in range(1, 4):
        result += 1
print(result)
```

A. 8
B. 9
C. 15
D. 20
E. an `int` other than 8, 9, 15, or 20

21) What does this print?

```
result = 0
i = 6
while i < 10:
    result += i
    i += 2
print(result)
```

A. 14
B. 20
C. 30
D. an `int` other than 14, 20, or 30
E. nothing: it doesn't print an `int`

22) What does this print?

```
i = 4
result = -1
while i >= 0:
    if (i + 1) % 2 == 0:
        result = i
    i += -1
print(result)
```

A. 0
B. 1
C. 2
D. 4
E. 5

23) What does this print?

```
s = 'apple'
i = 0
result = ''
while i < len(s):
    if s[i] == s[i + 1]:
        result += s[i]
    i += 1
print(result)
```

A. `p`
B. `pp`
C. `ale`
D. nothing: the final value of `result` is the empty string
E. nothing: the program crashes when it runs

24) What does this print?

```
s = 'mysterious'
i = 0
flag = False
while not flag:
    if s[i] in 'aeiou':
        flag = True
        i += 2
    else:
        i += 1
print(s[i])
```

A. `e`
B. `i`
C. `o`
D. `r`
E. nothing: the program loops forever and never reaches the `print` statement

25) What does this print?

```
n = 64
while n > 1:
    n = n / 2
print(n)
```

A. 0.0
B. 0.5
C. 1.0
D. 2.0
E. nothing: the program loops forever and never reaches the `print` statement

This is **Code Listing 1**, referred to in the next few question:

```
def print_n(s, n):        # line 1
    for i in range(n):    # line 2
        print(s)          # line 3

def f(n):                 # line 4
    if n % 2 == 0:        # line 5
        return n / 2      # line 6
    else:                 # line 7
        return 3 * n + 1  # line 8

def main():
    a = 3                 # line 9
    b = int(f(a + 1))     # line 10
    print_n('hello', b)   # line 11
```

Code Listing 1

26) In Code Listing 1, when `main()` is called, `'hello'` is printed twice.
A. True
B. False

27) In Code Listing 1, `main` has two local variables.
A. True
B. False

28) In Code Listing 1, if line 9 is changed to `a = 5`, then the program prints `'hello'` 4 times.
A. True
B. False

29) In Code Listing 1, `int(f(f(50)))` evaluates to 76.
A. True
B. False

30) In Code Listing 1, if line 2 was changed to `for i in range(1, n+1)`, then the program would print the same thing as if the change was not made.
A. True
B. False

31) In Code Listing 1, if function `main()` was moved to be defined before function `print_n()`, the program would print the same thing as if the change was not made.
A. True
B. False

32) In Code Listing 1, if lines 9, 10, and 11 had their indent removed so that they each started in the same column as the `d` in `def` on line 4, then calling `main()` would print `'hello'` twice.
A. True
B. False

33) Consider this code:

```
def reset(n):
    n = 0

def test1(x):
    x = 1
    reset(x)
    print(x)

def test2():
    n = 1
    reset(n)
    print(n)
```

i) Calling `test1(0)` prints 0.
ii) Calling `test2()` prints 0.

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

34) Suppose we want a function that takes a string `s` as input and returns a string as follows:

- If `s` *ends* with a newline character, then the returned string is the same as `s` except that the one newline at the end has been removed.
- If `s` *does not end* with a newline character, then the returned string is the same as `s`.

Here are two possible implementations of this function:

`def chop1(s):`
`    if s == '':`
`        return s`
`    elif s[-1] == '\n':`
`        return s[:-1]`
`    else:`
`        return s`
	`def chop2(s):`
	`    n = len(s)`
	`    if n == 0:`
	`        return s`
	`    elif s[n] == '\n':`
	`        return s[:n-1]`
	`    else:`
	`        return s`

A. both are **correct** implementations
B. both are **incorrect** implementations
C. `chop1` is a **correct** implementation, and `chop2` is an **incorrect** implementation
D. `chop1` is an **incorrect** implementation, and `chop2` is a **incorrect** implementation

35) Suppose this line of code correctly opens the non-empty text file named `data.txt`:

```
f = open('data.txt')
```

How can you print just the **first** line of `data.txt`?

A. `print(f[0])`
B. `print(f.read())`
C. `print(f.readline())`
D. all of the above

36) Suppose this line of code correctly opens the text file named `data.txt`:

```
f = open('data.txt')
```

Which statement prints the total number of characters in `data.txt` ?

A. `print(f)`
B. `print(len(f))`
C. `print(f.read())`
D. `print(len(f.read()))`

37) Suppose this line of code correctly opens the text file named `data.txt`:

```
f = open('data.txt')
```

A. `f` is open just for reading
B. `f` is open just for writing
C. `f` is open for both reading and writing

38) Which function returns the index location of an `int` `x` in a list `lst`?

A.
```
def linear_search1(x, lst):
    for i in range(len(lst) - 1):
        if lst[i] == x:
            return i
    return -1
```
B.
```
def linear_search2(x, lst):
    i = 0
    while i < len(lst):
        if lst[i] == x:
            return i
        i += 1
        return -1
```
C.
```
def linear_search3(x, lst):
    for i in lst:
        if i == x:
            return i
    return -1
```
D.
```
def linear_search4(x, lst):
    i = 0
    while i < len(lst):
        if lst[i] == x:
            return i
        i += 1
    return -1
```

39) What does this print?

```
def f(lst, target):
    for i in range(len(lst)):
        if lst[i] + 5 == target:
            return i
    return -1

data = [10, 3, 6, 5, 2, 7]
print(f(data, 6))
```

A. -1
B. 2
C. 5
D. 6
E. an int other than -1, 2, 5, or 6

40) Here are two possible implementations of a function that is meant to return the sum of the numbers in a list of numbers:

```
def sum1(lst):
    for n in lst:
        result += n
    return result
```

```
def sum2(lst):
    result = 0
    i = 0
    while i < len(lst):
        result += lst[i]
    return result
```

A. both are **correct** implementations
B. both are **incorrect** implementations
C. `sum1` is a **correct** implementation, and `sum2` is an **incorrect** implementation
D. `sum1` is an **incorrect** implementation, and `sum2` is a **incorrect** implementation

41) What does this print?

```
lst = [4, 2, 5, 3, 1]
m = lst[0]
for x in lst:
     if x < m:
        m += x
print(m)
```

A. 1
B. 4
C. 11
D. 15
E. an `int` other than 1, 4, 11, or 15

42) What value of `x` makes this program print 4?

```
lst = [2, x, 1, 1, 3, 1, 3]
print(lst.count(lst[1]))  # prints 4
```

A. 0
B. 1
C. 2
D. 3
E. an `int` other than 0, 1, 2, or 3

43) What does this print?

```
A = [2, 2, 1, 1, 2, 2]
B = [A.count(1), A.count(2)]
print(B.count(1) + B.count(2))
```

A. 0
B. 1
C. 2
D. 3
E. an int other than 0, 1, 2, or 3

44) If you run binary search on this list, what is the first value the search compares to the target?

```
[4, 5, 7, 9, 10, 11, 15, 16, 20]
```

A. 4
B. 9
C. 10
D. 11
E. an `int` other than 4, 9, 10, or 11

45) Consider these statements:

i) Linear search requires that the data it is searching be in sorted order.
ii) Binary search requires that the data it is searching be in sorted order.

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

46) In the worst case, about how many comparisons does **selection sort** do to sort a list of n ints?

A. n
B. $2n$
C. n^2
D. n^3
E. 2^n

47) In the worst case, about how many comparisons must be done to test if a list of n ints is in ascending sorted order?

A. n
B. $2n$
C. n^2
D. n^3
E. 2^n

48) What does this print?

```
count = 0
for i in range(10):
    if i % 2 == 0:
        count += 1
print(count)
```

A. 5
B. 6
C. 9
D. 10
E. an `int` other than 5, 6, 9, or 10

49) What does this print?

```
count = 0
for i in range(10):
    for j in range(15):
        count += 1
print(count)
```

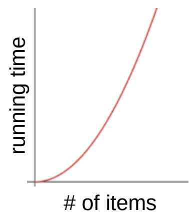
A. 23
B. 25
C. 126
D. 150
E. an `int` other than 23, 25, 126, or 150

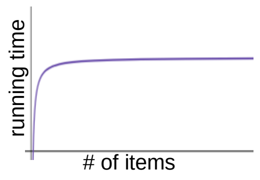
50) What does this print?

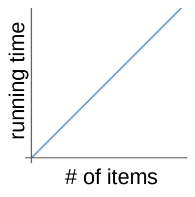
```
count = 0
for i in range(5):
    for j in range(5):
        if i != j:
            count += 1
print(count)
```

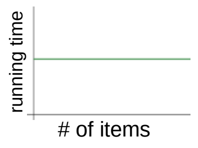
A. 5
B. 12
C. 20
D. 25
E. an `int` other than 5, 12, 20, or 25

51) Which graph best describes the worst-case running-time of the selection sort algorithm?
For each graph, the vertical y-axis is running time, and the horizontal x-axis is the number of items being sorted.

A. 

B. 

C. 

D. 

52) What is a recursive function?

A. a function that is called multiple times by other functions
B. a function that has no loops
C. a function that calls itself
D. a function that calls itself, and does not call any other functions

53) Consider these statements:

i) Any recursive function can be re-written as an equivalent function that doesn't use recursion.
ii) Any function that uses loops can be re-written as an equivalent function (or functions) that uses recursion instead of loops.

A. i) and ii) are both true
B. i) and ii) are both false
C. i) is true and ii) is false
D. i) is false and ii) is true

54) What does this print?

```
def g(n):
    if n <= 0:
        return 0
    else:
        return g(n - 2) + n

print(g(5))
```

A. 9
B. 10
C. 13
D. 15
E. an `int` other than 9, 10, 13, or 15

55) What does this print?

```
def h(n):
    if n == 0:
        return 0
    else:
        return h(n - 1)

print(h(100))
```

A. 0
B. 1
C. 99
D. 100
E. none of the above

56) What is pseudocode?

A. the generic name of the language that Python is automatically converted to just before it runs on a real computer
B. the generic name for any programming language, such as Python, that contains English words in it
C. a description of an algorithm/program designed for human reading
D. source code with one or more bugs in it

57) Which application is **NOT** a good fit for Python?

A. data science, e.g. processing and displaying data
B. machine learning scripting, e.g. processing data and running learning algorithms
C. high-performance real-time systems, such as airplane control software
D. back-end web development

58) What does this print?

```
lst = [1, 4, 3, 2, 5]
lst[1:4].sort()
lst.reverse()
print(lst[1] - lst[3])
```

A. -1
B. -2
C. 1
D. 2

59) What does this print?

```
numbers = [1, 2, 3, 4, 5]
numbers[2:3] = [6, 7]
print(numbers)
```

A. `[1, 2, 6, 7, 3, 4, 5]`
B. `[1, 2, [6, 7], 4, 5]`
C. `[1, 2, 6, 7, 4, 5]`
D. the code prints nothing due to an error

60) What does this print?

```
a, b, c, = 1, 'two', [3, 4]
c, a, b = b, c, a
print(2*a, 2*b, 2*c)
```

A. `[6, 8] 2 twotwo`
B. `[3, 4, 3, 4] 2 twotwo`
C. `[3, 4] 1 two`
D. the code prints something, but none of the above
E. the code prints nothing due to an error