CMPT 120 Standard Final Exam

Sample 2
Multiple Choice Questions

Duration	1 hour
Aids allowed	Pencil (or pen) and eraser. No notes, no papers, no books, no computers, no calculators, no cheat sheets, etc.
Scoring	For each question fill in **the one best answer** on the answer sheet. Each correct answer scores 1 point. Incorrect answers, multiple answers, illegible answers, or unanswered questions score 0 points.
During the exam	Raise your hand if you would like to speak with a proctor. Questions about exam/course content will **not** be answered during this exam.

1) What does this print?

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

A. 122
B. 342
C. 3332
D. 44442
E. nothing: the statement has an error

2) Consider this program:

```
a = -3
b = a - 3
a = b + a
b = a + b
print(a)  # line 1
print(b)  # line 2
```

i) line 1 prints -9
ii) line 2 prints -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 = 4
b = 1
???
print(a) # 1
print(b) # 4
```

Suppose `???` is replaced by one of the fragments below. Which one makes the code print 1 for `a` and 4 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

4) Consider this statement:

```
print(2 + (4 ??? 3))
```

How many of these 4 operators can replace **???** so that the statement prints 3?

```
+    -    *    %
```

A. 0
B. 1
C. 2
D. 3
E. 4

5) Consider these statements:

i) Python strings **can** be modified
ii) Python lists **cannot** 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[1 + lst[1]])
```

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

7) What does this print?

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

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

8) Consider this code:

```
s = 'thimble'
```

Which statement prints `himb` ?

A. `print(s[1:4])`
B. `print(s[1:5])`
C. `print(s[2:4])`
D. `print(s[2:5])`

9) What does this print?

```
d = {}
d[5] = 3
d[5] = 2
print(d[5])
```

A. 2
B. 3
C. it prints some value other than 2 or 3
D. this code crashes with an error when run

10) How many of these three programs print 6?

# program P	**# program Q**	**# program R**

`d = {'x':1, 'y':2, 'z':3}`
`total = 0`
`for x in range(len(d)):`
`    total += d[x]`
	`d = {'a':1, 'b':2, 'c':3}`
	`total = 0`
	`for x in d:`
	`    total += d[x]`
		`d = {'t':1, 's':2, 'u':3}`
		`total = 0`
		`for x in d:`
		`    total += x`

`print(total)`	`print(total)`	`print(total)`

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

11) What string values for `a` and `b` make this code print just the string `done`, and nothing else?

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

A.
```
a = 'cat'
b = 'dog'
```
B.
```
a = 'parrot'
b = 'dog'
```
C.
```
a = 'cat'
b = 'parrot'
```
D. There **are** strings values of `a` and `b` that make the code print just `done`, but none of the above options A, B, or C do that.
E. There are **no** possible string values for `a` and `b` that make the code print just `done`.

12) Which code fragment prints `good` *just* when the lengths of strings `a`, `b`, and `c` are *all* different, and `bad` in every other case?

A.
```
if len(a) != len(b) and len(b) != len(c):
    print('good')
else:
```

```
    print('bad')
```

B.
```
if not (len(a) == len(b) and len(b) == len(c)):
    print('good')
else:
    print('bad')
```

C.
```
if not (len(a) == len(b) or len(a) == len(c) or len(b) == len(c)):
    print('good')
else:
    print('bad')
```

D. All of A, B, or C
E. None of A, B, or C

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) - 1
        else:
            result += int(c)
    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) or (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 = 1
for i in range(5):
    if i > x:
       result += i
print(result)
```

A. 7
B. 8
C. 12
D. 13
E. it prints an `int` other than 7, 8, 12, or 13

17) What does this print?

```
s = 'orange'
result = ''
for i in s:
    if i < 'k':
```

```
        result += i
print(result)
```

A. the empty string: the final value of `result` is the empty string
B. `age`
C. `orn`
D. nothing: the program crashes when `i < 'k'` is evaluated
E. a string other than `age`, `orn`, or the empty string

18) What does this print?

```
lst = [4, 0, 9, 1]
result = 0
for i in range(len(lst)):
    result += lst[i] + 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 s in ['up', 'moose', 'elephant', '!']:
    if len(s) < len(result):
        result = s
print(result)
```

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

20) What does this print?

```
result = 0
for i in range(2, 5):
    for j in range(4):
        result += 1
```

```
print(result)
```

A. 4
B. 11
C. 12
D. 13
E. 16

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. it doesn't print an `int`

22) What does this print?

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

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

23) What does this print?

```
s = 'apple'
i = 1
result = '!'
while i < len(s):
```

```
    if s[i - 1] == s[i]:
        result += s[i]
    i += 1
print(result)
```

A. `!`
B. `!p`
C. `!pp`
D. `!pl`
E. nothing: the program crashes when run

24) What does this print?

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

A. `e`
B. `i`
C. `o`
D. `r`
E. nothing: the `print` statement is never called

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 `print` statement is never called

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

```
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('Kermit', b)  # line 11
```

Code Listing 1

26) In Code Listing 1, when `main()` is called, how many times is `Kermit` printed?

A. 0
B. 1
C. 2
D. 3
E. more than 3 times

27) In Code Listing 1, `main` has two local variables.

A. True
B. False

28) In Code Listing 1, how many times is `Kermit` printed if line 9 is changed to `a = 4` ?

A. 0
B. 1
C. 2

D. 3
E. more than 3

29) In Code Listing 1, `f(1) + f(4)` evaluates to 7.

A. True
B. False

30) In Code Listing 1, if line 2 was changed to `for i in range(2, n + 2)`, 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 start in the same column as the `d` in `def` on line 4, then calling `main()` would print `Kermit` 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 new 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[:len(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 **correct** implementation

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

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

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

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

36) Suppose this code correctly opens the text file named `animals.txt`:

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

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

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

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

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

`f` is open:

A. just for reading
B. just for writing
C. for both reading and writing
D. neither reading nor writing

38) Which function returns the index location of the `int x` in a list `lst`? Assume `x` occurs exactly once in `lst`.

A.

```
def search1(x, lst):
    for i in range(len(lst) - 1):
        if lst[i] == x:
            return i
    return -1
```

B.

```
def search2(x, lst):
    i = 0
    while i < len(lst):
        if lst[i] == x:
            return i
        i += 1
        return -1
```

C.

```
def search3(x, lst):
    for i in lst:
        if i == x:
            return i
    return -1
```

D.

```
def search4(x, lst):
    i = 0
    while i < len(lst):
        if lst[i] == x:
            return i
        i += 1
    return -1
```

E. none of the above

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

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

```
def addem1(lst):
    for n in lst:
        result += n
    return result
```

```
def addem2(lst):
    result = 0
    i = len(lst) - 1
    while i >= 0:
        result += lst[i]
    return result
```

A. both are **correct** implementations
B. both are **incorrect** implementations
C. `addem1` is a **correct** implementation, and `addem2` is an **incorrect** implementation
D. `addem1` is an **incorrect** implementation, and `addem2` is a **correct** implementation

41) What does this print?

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

A. 0
B. 1
C. 2
D. 10
E. an `int` other than 0, 1, 2, or 10

42) What value of `x` makes this program print 3?

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

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

43) What is the biggest number that this code prints?

```
for x in [0, 1, 2, 3, 4]:
    A = [x, 2, 1, 1, 2, 2]
    A[4] = x
    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` bigger than 3

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

```
[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) What is the minimum number of comparisons (i.e. calls to `<`) needed to check if a list of 50 different numbers is in ascending sorted order?

A. 48
B. 49
C. 50
D. 51

48) What does this print?

```
x = 0
for i in range(1, 100):
    if i % 2 == 0:
        x += 1
print(x)
```

A. 49
B. 50
C. 51
E. an `int` other than 49, 50, or 51

49) What does this print?

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

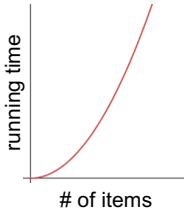
A. 23
B. 25
C. 126
D. 150
E. an `int` other than 23, 25, 126, or 150

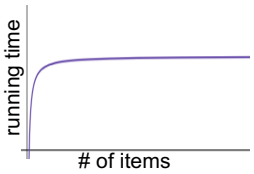
50) What does this print?

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

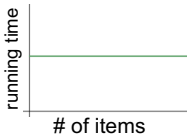
A. 5
B. 12
C. 20
D. 25
E. an `int` other than 5, 12, 20, or 25

51) Which graph best describes the worst-case running-time of the **selection sort** algorithm?

A.


B.


C.


D.


52) What is a recursive function? A function that:

A. does not called any other functions
B. has no loops
C. calls itself
D. 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 (or functions) 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(g(3)))
```

A. 4
B. 5
C. 6
D. an `int` other than 4, 5, or 6
E. nothing: it never returns

55) What does this print?

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

print(h(99))
```

A. 0
B. 1
C. 98
D. 99
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 a **BAD** 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 = [4, 1, 3, 2, 5]
lst = lst[1:4] + lst[:2]
lst.sort()
lst.reverse()
print(lst[1] - lst[3])
```

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

59) What doe this print?

```
s = 'abcde'
x = s[2:5] + s[1:4] + s[:3]
print('dab' in x)
print('deb' in x)
```

A.
```
False
False
```
B.
```
False
True
```
C.
```
True
False
```
D.
```
True
True
```

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] 4 twotwo`
C. `[3, 4] 1 two`
D. the code prints something, but none of the above
E. nothing: the code has an error