CMPT 120 Standard Final Exam

Sample 3
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('1' * 2 + '3')
```

A. 5
B. 113
C. 123
D. 223
E. nothing: the statement has an error

2) Which program prints the same thing as this one?

```
a = 2
b = 2
a = b - a
b = b - a
print(a)
print(b)
```

A.
```
a = 2
b = 2
a = a + b
b = a - b
print(a)
print(b)
```

B.
```
a = 2
b = 2
a = b - a
b = b + a
print(a)
print(b)
```

C.
```
a = 2
b = 2
b = b + a
b = b - a
print(a)
print(b)
```

D.
```
a = 2
b = 2
b = a - b
a = a + b
print(a)
print(b)
```

3) Assuming `a` is initialized 2 and `b` is initialized to 6, which code fragment prints 6 on one line, and then 2 on the next line?

A.
```
tmp = a
b = a
a = tmp

print(a)
print(b)
```

B.
```
tmp = b
a = b
b = tmp

print(a)
print(b)
```

C.
```
tmp = b
b = a
b = tmp

print(a)
print(b)
```

D.
```
tmp = a
a = b
b = tmp

print(a)
print(b)
```

4) Consider this statement:

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

How many of these 4 arithmetic operators could replace `???` so that it prints 3?

```
+    -    *    %
```

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

5) Consider these statements:

i) You *can* change the length of a Python string
ii) You *can* change the length of a Python list

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, 3]
print(lst[lst[2]] + lst[lst[3]])
```

A. 1
B. 2
C. 3
D. some `int` other than 1, 2, or 3
E. nothing: there is an error

7) What does this print?

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

A. `[2, 0, 3]` B. `[0, 1, 3]` C. `[2, 1, 3]` D. nothing: the code has an error

8) What does this print?

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print(len(lst[2:5]))
```

A. 5 B. 4 C. 3 D. 2

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. it 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]`
`print(total)`
	`d = {'a':1, 'b':2, 'c':3}`
	`total = 0`
	`for x in d:`
	`  total += d[x]`
	`print(total)`
		`d = {'t':1, 's':2, 'u':3}`
		`total = 0`
		`for x in d:`
		`  total += x`
		`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** string values of `a` and `b` that make the code print just `done`, but none of 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) Suppose `x`, `y`, and `z` are `int` variables (but we don't know which `int`s exactly). Consider these statements:

i) if `[x, y, z] == [y, y, y]` evaluates to `True`, then `x`, `y`, and `z` all equal the same value

ii) if `x`, `y`, and `z` all equal the same value, then `[x, y, z] == [y, y, y]` evaluates to `True`

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

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

14) What function call returns the same value as `f('0')` ?

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

    return total
```

A. `f('1')` B. `f('2')` C. `f('3')` D. none of A, B, or C

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

```
(a < b) or (b < a)
```

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?

```
cutoff = 10
result = 0
for i in range(20):
    if i < cutoff:
       result += 1
print(result)
```

A. 9 B. 10 C. 11 D. 19 E. 20

17) What does this print?

```
s = 'soccer'
result = 0
for i in range(len(s)):
    if s[i] <= s[i + 1]:
        result += 1
print(result)
```

A. 0 B. 1 C. 2 D. 3 E. nothing: the program crashes due to an error

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. 15 D. 20

19) What value of `lst` makes this print `!` ?

```
result = 'start'
for s in lst:
    if len(s) < len(result):
        result = s
print(result)
```

A. `['!', 'up', 'moose', 'elephant']`
B. `['up', '!', 'moose', 'elephant']`
C. `['up', 'moose', '!', 'elephant']`
D. `['up', 'moose', 'elephant', '!']`
E. all of A, B, C, and D

20) What does this print?

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

A. 12 B. 13 C. 16 D. 20

21) Which program prints the biggest number?

```
# program 1
result = 0
i = 0
while i < 5:
    i += 1
    result += i
print(result)
```

```
# program 2
result = 0
i = 0
while i < 5:
    result += i + 1
    i += 1
print(result)
```

A. program 1 prints the biggest number
B. program 2 prints the biggest number
C. they print the same number

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?

```
a = 1
b = 20
while a * a <= b:
    a += 1
print(a)
```

A. 20 B. 19 C. 6 D. 5 E. nothing: the `print` statement is never called

25) What does this print?

```
result = 0
i = 0
while i < 3:
    j = 0
    while j < 4:
        result += 1
        j += 1
    i += 1
print(result)
```

A. 6 B. 8 C. 12 D. 20 E. nothing: the `print` statement is never called

26) This program is meant to print the sum of the numbers in `L`, but it may have a bug. When the program runs, what line causes a *run-time error*?

```
L = [5, 1, 3]
i = 0                      # line 1
total = 0
while i <= len(L):         # line 2
   total = L[i] + total    # line 3
   i = i + 1               # line 4

print(total)
```

A. line 1 B. line 2 C. line 3 D. line 4 E. there is no error: the program correctly prints the sum of any list L of numbers

27) What does this print?

```
a = 1
b = 2

def f(a):
    a = a + a
    a += a
    print(a)

f(b)
```

A. 8 B. 4 C. 2 D. nothing: there is an error in the program

28) What does this print?

```
def f(n - 1):
    return n + 1

print(f(0) + f(1))
```

A. -1 B. 0 C. 1 D. 2 E. nothing: the program has an error

29) Consider this program:

```
x = 1
y = 4
x = x - y
y = y + x
???
print(x)
print(y)
```

What can replace `???` to make it print:

```
4
1
```

A. `y = y - x` B. `x = x - y` C. `x = y - x` D. `y = x + y`

30) What does this print?

```
def f(n):
    result = n - 1
    for i in range(2, n + 1):
        result += i
    return result

print(f(3))
```

A. 5 B. 6 C. 7 D. 8

31) What does this print?

```
def greet(s):
    say_hi(s)

def say_hi(s):
    print("Hi " + s + "!")

greet("Alice")
```

A. `Hi Alice!` B. it runs without error and prints something other than `Hi Alice!` C. the program has an error and so prints nothing

32) If C is a temperature in Celsius, then this formula converts it to temperature F in Fahrenheit:

$$F = \left(\frac{9}{5}\right)C + 32$$

This function correctly converts Celius to Fahrenheit:

```
def to_fahrenheit(C):
    return (9/5)C + 32
```

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)]
    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 the non-empty text file `errors.txt` is opened correctly by the program below. What gets printed?

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

A. the first character of the file

B. the first line of the file

C. the entire file

D. nothing: the program has an error

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

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

What does the program print?

A. the number of lines in the file

B. the number of characters in the file

C. the size of the file in bytes

D. nothing: the code has an error

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

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

`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 always 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
```

C.
```
def search3(x, lst):
    i = 0
    while i < len(lst):
        if lst[i] == x:
            return lst[i]
        i += 1
    return -1
```

B.
```
def search2(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
```

39) What does this print?

```
def f(lst):
    result = 0
    for i in range(1, len(lst)):
        if lst[i-1] < lst[i]:
            result = lst[i]
    return result

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

A. 2 B. 3 C. 5 D. 6 E. 7

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

```
def addem1(lst):
    result = 0
    for i in range(len(lst)):
        result += lst[i]
    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]
acc = 0
for x in lst:
    if x > lst[0]:
        acc += x
print(acc)
```

A. 4 B. 5 C. 9 D. 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 last number that this code prints?

```
for x in [1, 2, 4, -1]:
    A = [x, x + 1, 2, 3]
    B = [A.count(1), A.count(x)]
    print(B[0] + B[1])
```

A. 1 B. 2 C. 3 D. 4 E. 5

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

```
[4, 5, 6, 10, 11, 12, 13, 16, 21]
```

A. 4 B. 10 C. 11 D. 12 E. an `int` other than 4, 10, 11, or 12

45) Suppose `L` is a list of the numbers from 1, 2, …, 99, 100 in some random order, and linear search is used to determine if a given target number `x` is in `L`.

i) The fewest number of comparisons linear search might do to is 1 comparison.
ii) The greatest number comparisons linear search might do to is 99 comparisons.

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) Marge has a bag filled with 5 marbles numbered 1, 2, 3, 4, 5. She repeats these steps until the bag is empty:

- She removes the *lowest-numbered* marble from the bag.
- She places it on the right end of the marbles already on the table.

When there are n marbles in the bag it takes Marge exactly n seconds to remove the smallest and place it on the table.

How many seconds in total would it take Marge to remove and place all 5 marbles?

A. 5 B. 10 C. 12 D. 15 E. 20

48) What does this print?

```
x = 0
for i in range(10, 20):
    if i % 2 == 1:
        x += 1
print(x)
```

A. 4 B. 5 C. 6 E. an `int` other than 4, 5, or 6

49) What does this print?

```
x = 0
for i in range(5):
    for j in range(5):
        x += 2
print(x)
```

A. 16 B. 25 C. 32 D. 50

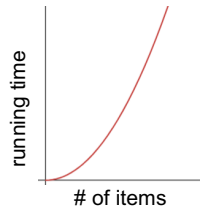
50) What does this print?

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

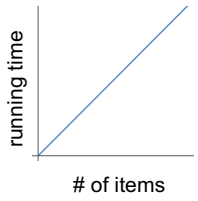
A. 9 B. 10 C. 18 D. 20 E. 90

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

A.

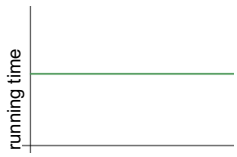


C.



B.

D.



52) A function is recursive if it:

A. has no loops
B. does not call any other functions
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 - 1) + 2

print(g(3) + g(4))
```

A. 6 B. 8 C. 14 E. nothing: `g` never returns

55) What does this print?

```
def h(n):
    if n <= 2:
        return n
    else:
        return h(n) - 1

print(h(4))
```

A. -1 B. 0 C. 2 D. nothing: `h(4)` does not return a value

56) What is pseudocode?

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

57) Consider these statements:

i) Python is a good language for implementing a high-performance real-time system, such as an operating system of 3D graphics systems
ii) Python is a good language for running (but not necessarily implementing) machine learning algorithms

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

58) What does this print?

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

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

59) What does this print?

```
s = 'abcde'
for i in range(3):
    s = s[1:4] + s[:2]
print(s)
```

A. `abcde` B. `dabcd` C. `eabcd` D. `bcdab`

60) What does this print?

```
a, b, c = 1, 'two', [3, 4]
c, a, b = a, c, b
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 program has an error