

CPSC 217 Final Examination

Duration: 120 minutes

24 April 2009

- This exam has 70 questions and 15 pages.
- This exam is closed book. No notes, books, calculators or electronic devices, or other assistance may be used.
- Mark your answers on the supplied answer sheet.
- Assume numbers are base ten unless stated otherwise.
- Assume questions refer to Python unless stated otherwise.
- If you think multiple answers may be correct, choose the best answer.

Part 1

1. TRUE/FALSE: 'a' != 'b' or 1 == 2
2. TRUE/FALSE: (2 + 3) * 5 == 2 + 3 * 5
3. TRUE/FALSE: True and False
4. TRUE/FALSE: not(a or b) != (not a) and (not b)
5. TRUE/FALSE: not(not(not True))
6. TRUE/FALSE: One advantage of modules is code reuse.
7. TRUE/FALSE: A module cannot be tested separately from the code that imports it.
8. TRUE/FALSE: A greedy algorithm will always find a solution that is best overall.
9. TRUE/FALSE: A brute-force algorithm will try all possible solutions until it finds one that works.
10. TRUE/FALSE: The print statement is a debugging aid.
11. TRUE/FALSE: Black-box testing requires examining a program's code.
12. TRUE/FALSE: All program designs can be expressed using the I-P-O model.
13. TRUE/FALSE: A program can have multiple files open at a time.

Part 2

Use this definition for the questions in this part:

$M = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]$

14. What is `len(M)`?

- (A) 0
- (B) 1
- (C) 3
- (D) 9
- (E) 42

15. What is `M[2]`?

- (A) 2
- (B) 3
- (C) [4, 5, 6]
- (D) [7, 8, 9]

16. What is `M[2][3]`?

- (A) 5
- (B) 6
- (C) 8
- (D) 9
- (E) An error

17. What is `M[1][2]`?

- (A) 2
- (B) 5
- (C) 6
- (D) 8
- (E) An error

18. What does this code print when run?

```
x = 0
for i in range(3):
    x = x + M[i][i]
print x
```

- (A) 0
- (B) 15
- (C) 19
- (D) 45
- (E) 54

19. What does this code print when run?

```
x = 0
for i in range(3):
    for j in range(3):
        x = x + M[i][j]
print x
```

- (A) 0
- (B) 15
- (C) 19
- (D) 45
- (E) 54

20. What does this code print when run?

```
x = 0
for i in range(3):
    for j in range(i):
        x = x + M[i][j]
print x
```

- (A) 0
- (B) 15
- (C) 19
- (D) 45
- (E) 54

Part 3

Use these definitions for the questions in this part:

L = [1, 2, 3, 4]

T = (0, 1, 2, 3, 4)

D = { 0: 1, 2: 3, 4: 5, 6: 7 }

- 21. TRUE/FALSE: L[-1] == T[-1]
- 22. TRUE/FALSE: len(L[1:2]) == 2
- 23. TRUE/FALSE: D[4] == L[1] + T[2] + D[0]
- 24. TRUE/FALSE: T[:2] == [0, 1]
- 25. TRUE/FALSE: D[6] != 7
- 26. TRUE/FALSE: T[3] == 3

Part 4

Use the following code for this part.

```
def foo(m, n):
    if m == 0:
        return n + 1
    elif m > 0 and n == 0:
        return foo(m-1, 1)
    elif m > 0 and n > 0:
        return foo(m-1, foo(m, n-1))
```

foo(1, 0)

27. How many calls are made to the function foo when this code is run?

- (A) 1
- (B) 2
- (C) 3
- (D) 4
- (E) 5

28. What value would foo(1, 1) return?

- (A) 1
- (B) 2
- (C) 3
- (D) 4
- (E) 5

Part 5

This code is supposed to print the lines of a specified input file backwards. The input file name is given as a command-line argument.

```
import sys
AAA

if len(sys.argv) BBB:
    print 'Needs input file name'
    sys.exit(1)

f = open(CCC, DDD)

L = []
for line in f:
    L.append(line)
EEE

i = len(L)
while FFF:
    GGG
```

29. What should go in the spot labeled AAA?
- (A) `import sys`
 - (B) `import math`
 - (C) `import open`
 - (D) `import file`
 - (E) Nothing
30. What should go in the spot labeled BBB?
- (A) `!= 1`
 - (B) `!= 2`
 - (C) `== 1`
 - (D) `== 2`
 - (E) `> 2`
31. What should go in the spot labeled CCC?
- (A) `sys.argv[0]`
 - (B) `sys.argv[1]`
 - (C) `sys.argv[2]`
 - (D) `sys.argv`
32. What should go in the spot labeled DDD?
- (A) `'a'`
 - (B) `'r'`
 - (C) `'r+'`
 - (D) `'w'`
33. What should go in the spot labeled EEE?
- (A) `f.close()`
 - (B) `close(f)`
 - (C) `release(f)`
 - (D) `unopen(f)`
34. What should go in the spot labeled FFF?
- (A) `i >= 0`
 - (B) `i > 0`
 - (C) `i > 1`
 - (D) `i < len(L)`
 - (E) `i <= len(L)`
35. What should go in the spot labeled GGG?
- (A) `print i`
 - (B) `print line`
 - (C) `print L[i],`
`i = i - 1`
 - (D) `i = i - 1`
`print L[i],`
 - (E) `i = i + 1`
`print L[i],`

Part 6

36. `checkpassword` is a function that takes an integer argument as a password. It returns `True` if the integer password is correct and `False` if the integer password is wrong. Consider the following code:

```
MAXPASSWD = 12345
i = 0
while i < MAXPASSWD:
    if checkpassword(i):
        print 'password is', i
        break
    i = i + 1
```

This is an example of a

- (A) brute-force algorithm
 - (B) fuzzing algorithm
 - (C) greedy algorithm
 - (D) knapsack algorithm
 - (E) partitioning algorithm
37. What does the following code print when run?
- ```
try:
 print 'A',
 int('xxx')
 print 'B',
except:
 print 'C',
```
- (A) A
  - (B) A B
  - (C) A C
  - (D) A B C
  - (E) Nothing – there is an error in the code
38. A brute-force algorithm might be sped up by
- (A) reordering the search space
  - (B) pruning the search space
  - (C) sorting the search space from smallest to largest
  - (D) A & B only
  - (E) A, B, & C
39. 109 is a base ten number. What is it in octal?
- (A) 81
  - (B) 155
  - (C) 551
  - (D) 968
  - (E) 1550

40. 9A is a hexadecimal number. What is it in base 10?

- (A) 19
- (B) 144
- (C) 145
- (D) 154
- (E) 291

41. 111 is a base one number. In base ten it is

- (A) 2
- (B) 3
- (C) 7
- (D) 111
- (E) nothing – there's no such thing as base one

42. 3243 is a base ten number. In base 16 it is

- (A) BAC
- (B) CAB
- (C) CAD
- (D) DAB
- (E) 12867

43. The two programs below are run using

```
python a.py | python b.py
```

```
a.py
for i in range(5):
 print i, i ** 2
```

```
b.py
x = 0
while x < 6:
 s = raw_input()
 fields = s.split()
 x = x + int(fields[0])
print fields[1]
```

What is the output?

- (A) 2
- (B) 3
- (C) 4
- (D) 9
- (E) 16

44. Consider the following module:

```
def foo():
 print 'Hello, world!'
```

You want the function `foo` to be called when this module is imported. You need to add at the end of the module

- (A) Nothing
- (B) `foo`
- (C) `foo()`
- (D) `if __name__ == '__main__':  
 foo()`

45. Module `foo` contains

```
print 'X'
```

How many Xs are printed by

```
import foo
import foo
import foo
```

- (A) 0
- (B) 1
- (C) 2
- (D) 3

46. What does the code below print when run?

```
i = 0
while i < 7:
 i = (i + 1) % 5
 print i,
```

- (A) 1 2 3 4 0 1 2 3 4 0...
- (B) 0 1 2 3 4 0 1 2 3 4...
- (C) 0 1 2 3 4 5 0 1 2 3 4 5...
- (D) 1 2 3 4 5 0 1 2 3 4 5 0...
- (E) 1 2 3 4 5 6

47. How many bits are in one kilobyte?

- (A) 1024
- (B) 2048
- (C) 4096
- (D) 8192
- (E) 16384



48. How many bits are necessary to represent a single uppercase letter in the alphabet?
- (A) 1
  - (B) 5
  - (C) 8
  - (D) 16
  - (E) 32
49. The ASCII representation of the letter “q” is 71 in hexadecimal. If you interpreted that as a 7-bit 2’s complement number, what would it be?
- (A) -71
  - (B) -15
  - (C) -14
  - (D) 15
  - (E) 71
50. What does the code below print when run?

```
def T2(x):
 return x * 2
def dozen():
 return T2('12')
def enestrate():
 return T2(3)
print int(dozen()) + enestrate()
```

- (A) 15
  - (B) 18
  - (C) 36
  - (D) 1218
  - (E) Nothing – an error occurs
51. How many function calls occur when this code is run?

```
def A():
 B()
 C()
def B():
 C()
def C():
 return
 B()
A()
```

- (A) 2
- (B) 3
- (C) 4
- (D) 5
- (E) An infinite number

52. What is the value of x after this code is run?

```
x = 12
def A():
 global x
 x = 5
def B():
 x = 7
A()
B()
```

- (A) 5
- (B) 7
- (C) 12
- (D) An error occurs when it is run

53. A design method involving repeated decomposition is called

- (A) bottom-up design
- (B) cemetery design
- (C) middle-out design
- (D) top-down design
- (E) white-box design

54. What is the value of L after this code is run?

```
L = [1, 2, 3]
def A():
 global L
 L[1] = 4
def B():
 L[2] = 2
A()
B()
```

- (A) [1, 2, 3]
- (B) [1, 2, 4]
- (C) [1, 4, 2]
- (D) [1, 4, 3]
- (E) An error occurs when it is run

55. What is printed when this code is run?

```
def A(x):
 x = x + 5
def B(x):
 x = 12
x = 76
B(x)
A(x)
print x
```

- (A) 12
- (B) 17
- (C) 76
- (D) 81
- (E) Nothing – an error occurs when it is run

56. What is printed when this code is run?

```
x = 5
if x < 7:
 print x,
 x = 23
if x > 7:
 print x,
else:
 print x + 1
```

- (A) 5
- (B) 23
- (C) 5 23
- (D) 5 6
- (E) 5 24

57. What is printed when this code is run?

```
x = 5
if x < 7:
 print x,
 x = 23
elif x > 7:
 print x,
else:
 print x + 1
```

- (A) 5
- (B) 23
- (C) 5 23
- (D) 5 6
- (E) 5 24

58. Consider the following code.

```
NTRIES = 3
MAX = 10
MIN = 1
try = 0
while try < NTRIES:
 try = try + 1
 s = raw_input()
 n = int(s)
 if MIN <= n and n <= MAX:
 break
```

When run, this code

- (A) gives the user two tries to enter a number between 1 and 10
  - (B) gives the user three tries to enter a number between 1 and 10
  - (C) gives the user four tries to enter a number between 1 and 10
  - (D) gives the user multiple tries to enter a number between 0 and 9
  - (E) does nothing – there is an error
59. What does this code draw when run?

```
import turtle
for i in range(10):
 turtle.forward(50)
 turtle.left(60)
```

- (A) Pentagon
  - (B) Hexagon
  - (C) Octagon
  - (D) Nonagon
  - (E) Decagon
60. This code is supposed to print out the string `s` backwards.

```
s = 'abcde'
t = ''
for ch in s:
 t = XXX
print t
```

What goes in the spot marked `XXX`?

- (A) `ch + t`
- (B) `t + ch`
- (C) `ch`
- (D) `s`

61. What does this code print when run?

```
L = [0, 1, 2, 3]
for i in L:
 print i,
 L.append(42)
```

- (A) 0 1 2 3
- (B) 0 1 2 3 42
- (C) 0 1 2 3 42 42 42 42
- (D) 0 1 2 3, followed by an infinite number of 42s
- (E) Nothing – there is a syntax error

62. What does this code print when run?

```
L = (0, 1, 2, 3)
L.append(42)
print L
```

- (A) (0, 1, 2, 3)
- (B) (0, 1, 2, 3, 42)
- (C) (42,)
- (D) (42, 0, 1, 2, 3)
- (E) Nothing – there is an error

63. What does this Python program print, when run as `python foo.py`?

```
import sys
print 'python', sys.argv[0]
```

- (A) python python
- (B) python sys.argv[0]
- (C) foo.py
- (D) python foo.py
- (E) Something else not listed above

64. What does this code print when run?

```
ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
L = [8, 1, 12]
for i in L:
 print ALPHA[i],
```

- (A) H A L
- (B) I B M
- (C) J C N
- (D) 8 1 12
- (E) Nothing – there is an error

65. The `while` statement in Python has an optional `else` part – this is code which is run if the `while` loop is *not* exited with `break`. Consider the following code.

```
L = (1, 3, 5, 7)
i = 0
key = 4
while i < len(L):
 if key == L[i]:
 print 'key found'
 break
 i = i + 1
else:
 print 'key not found'
```

What does this code print when run?

- (A) `key found`
  - (B) `key not found`
  - (C) `key found`, followed by `key not found`
  - (D) Nothing – there is an error
66. `spell` is a program that prints misspelled words in a file to its output, one misspelled word per line. What does the Unix pipeline below do?

```
spell foo | sort | uniq
```

- (A) Prints misspelled words in `foo`
  - (B) Prints misspelled words in `foo`, sorted
  - (C) Prints misspelled words in `foo`, sorted; multiple occurrences of a misspelled word are shown only once
67. To extract the second column of a data file with tab-separated fields, you would use
- (A) The `col` command with option `-f1`
  - (B) The `col` command with option `-c2`
  - (C) The `cut` command with option `-f1`
  - (D) The `cut` command with option `-f2`
  - (E) The `cut` command with option `-c2`

68. The wildcard `*b*.py` selects

- (A) Files containing a `b` and ending in `.py`
- (B) Files containing both `b` and `.py`
- (C) Files containing a `b`
- (D) Files ending in `.py`
- (E) Files containing `.py`

69. Modules should exhibit

- (A) low cohesion and high coupling
- (B) high cohesion and high coupling
- (C) high cohesion and low coupling
- (D) low cohesion and low coupling

70. **BONUS:** The name of the ghost with gout had a silent letter at the end. What was it?

- (A) e
- (B) j
- (C) q
- (D) w
- (E) z