

CS 61A Midterm 1

Raymond Tsao

TOTAL POINTS

26.9995 / 30

QUESTION 1

What Would Python Display? 4 pts

1.1 (3 and 4) - 5 **0.9999 / 1**

+ 0 pts Incorrect: -2

✓ + 1 pts Correct: -1

+ 0 pts Incorrect: 1

+ 0 pts Incorrect: 2

+ 0 pts Incorrect: Error

+ 0 pts Blank

✓ - 0.0001 pts Bubble not filled fully

1.2 oski(abs) 1 / 1

+ 0 pts Incorrect: -2

+ 0 pts Incorrect: -1

+ 0 pts Incorrect: 1

✓ + 1 pts Correct: 2

+ 0 pts Incorrect: Function

+ 0 pts Incorrect: Error

+ 0 pts Blank

- 0.0001 pts Bubble not filled fully

1.3 print(print(print(s, s) or print("Who's There?")), "Who?") 2 / 2

✓ + 2 pts Correct:

`

Knock Knock

Who's There?

None

None Who?

`

`Knock` can be `Boom`, `Thump`, `Ring`, or `Buzz`

+ 0 pts Misread strings (e.g. forgot question mark, incorrect capitalization)

- 1 pts One line incorrect/missing

- 1 pts `None` line swapped with `None Who?`

- 1 pts Extra commas/quotes

+ 0 pts Incorrect (2 or more lines wrong) / blank

QUESTION 2

Don't Call Me, I'll Call You 12 pts

2.1 Blank (a) 1 / 1

✓ + 1 pts Correct: 2

+ 0 pts Incorrect/Blank

2.2 Blank (b) 1 / 1

✓ + 1 pts Correct: 1

+ 0 pts Incorrect/Blank

2.3 Blank (c) 1 / 1

+ 0 pts Incorrect: `Global`

+ 0 pts Incorrect: `f`

✓ + 1 pts Correct: `f1`

+ 0 pts Blank

- 0.0001 pts Bubble not filled fully

2.4 Blank (d) 0.9999 / 1

+ 0 pts Incorrect: `Global`

+ 0 pts Incorrect: `f`

✓ + 1 pts Correct: `f1`

+ 0 pts Blank

✓ - 0.0001 pts Bubble not filled fully

2.5 Blank (e) 2 / 2

✓ + 1 pts Answer is a lambda `function`

✓ + 1 pts `[parent=Global]` (only applies if item 1 is met)

+ 0 pts Incorrect/Blank

2.6 Blank (f) 1 / 1

- ✓ + 1 pts Correct: 3
- + 0 pts Incorrect/Blank

2.7 Blank (g) 0.9999 / 1

- ✓ + 1 pts Correct: `Global`
- + 0 pts Incorrect: `f`
- + 0 pts Incorrect: `zero`
- + 0 pts Incorrect: `f1`
- + 0 pts Incorrect: `f2`
- + 0 pts Blank
- ✓ - 0.0001 pts Bubble not filled fully

2.8 Blank (h) 1 / 1

- ✓ + 1 pts Correct: 1
- + 0 pts Incorrect/Blank

2.9 Blank (i) 1 / 1

- ✓ + 1 pts Correct: 10
- + 0 pts Incorrect/Blank

2.10 Explain the Error 2 / 2

- ✓ + 2 pts Correct:
max was reassigned to a number (1)
 - + 1.5 pts Partial: Some statement that `max` was reassigned to an integer, but either interpreting the result as `y(x * x, 3 * x)` (ex. "max is replaced with y") or wrong integer value of `max` specified
 - 0.5 pts Answer is 20 words or longer
- + 0 pts Incorrect/Blank

QUESTION 3

It's Perfect 10 pts

3.1 Blank 1 0.9999 / 1

- ✓ + 1 pts Correct: `k < n``
- + 0 pts Incorrect: `k <= n``
- + 0 pts Incorrect: `k < total``
- + 0 pts Incorrect: `k <= total``
- + 0 pts Blank
- ✓ - 0.0001 pts Bubble not filled fully

3.2 Blank 2 1 / 1

- ✓ + 1 pts Correct: `n % k == 0`` or equivalent
- + 0 pts Incorrect/Blank

3.3 Blank 3 1 / 1

- ✓ + 1 pts Correct: `total = total + k``
- + 0 pts Incorrect/Blank

3.4 Blank 4 1 / 1

- ✓ + 1 pts Correct: `total <= n``, `total < n``, or equivalent
- + 0 pts Incorrect/Blank

3.5 Blank 5 2 / 2

- ✓ + 2 pts Correct: `classify(n+k) == "perfect"`
 - 0.5 pts Minor Syntax Error
 - + 0 pts Incorrect/Blank
 - + 1 pts Partial: Called `classify`` correctly to check if some number was perfect
 - + 1 pts Partial: Called `classify(n+k)`` (and no additional conditions)

3.6 Blank 6 0.9999 / 1

- + 0 pts Incorrect: `n = n + k``
- + 0 pts Incorrect: `n = n + 1``
- + 0 pts Incorrect: `n = n - k``
- + 0 pts Incorrect: `n = n - 1``
- ✓ + 1 pts Correct: `return n+k``
- + 0 pts Incorrect: `return n``
- + 0 pts Incorrect: `return k``
- + 0 pts Blank
- ✓ - 0.0001 pts Bubble not filled fully

3.7 Blank 7 1 / 1

- ✓ + 1 pts Correct: `k > 0``
- Alternate Solution: `classify(n-k) == "perfect"`
 - 0.25 pts Minor Syntax Error
 - + 0 pts Incorrect/Blank

3.8 Blank 8 2 / 2

- + 2 pts Correct:
 - `k = -k + 1`` or
 - `k = abs(k) + 1``

✓ + 2 pts Alternate Solution

``k = k + 1``

+ 0 pts Incorrect/Blank

QUESTION 4

Super Powers 4 pts

4.1 Correct funsquares 0 / 2

✓ - 1 pts Incorrect: A

- 1 pts Incorrect: B

- 1 pts Incorrect: C

✓ + 1 pts Correct: D

✓ - 1 pts Incorrect: E

✓ + 1 pts Correct: F

+ 0 pts Blank

4.2 quadruple 1 / 1

✓ + 1 pts Correct: ``lambda x: 2 * x``

+ 0.5 pts Syntax Incorrect: ``lambda x: 2x`` or ``lambda`

`x: 2 * x``

+ 0 pts Incorrect/Blank

4.3 funfourth 0 / 1

+ 1 pts Correct

``funsquare(funsquare)``

+ 0.5 pts Missing closing parentheses

✓ + 0 pts Incorrect/Blank

QUESTION 5

Busy Beaver 0 pts

5.1 Blank 1 0 / 0

+ 0 pts Correct: ``f``

+ 0 pts Incorrect: ``g``

+ 0 pts Incorrect: ``f()``

+ 0 pts Incorrect: ``g()``

✓ + 0 pts Blank/Multiple Selected

+ 0 pts Bubble not fully filled

5.2 Blank 2 0 / 0

+ 0 pts Correct:

``lambda f: lambda: f() or f()``

✓ + 0 pts Incorrect/Blank

5.3 Blank 3 0 / 0

+ 0 pts Correct: ``()``

+ 0 pts Incorrect: ``f)``

+ 0 pts Incorrect: ``g)``

✓ + 0 pts Blank

+ 0 pts Bubble not fully filled

INSTRUCTIONS

You have 1 hour and 50 minutes to complete the exam.

- The exam is closed book, closed notes, closed computer, closed calculator, except one 8.5" × 11" page of your own creation and the provided midterm 1 study guide.
- Mark your answers on the exam itself in the spaces provided. We will not grade answers written on scratch paper or outside the designated answer spaces.
- If you need to use the restroom, bring your phone and exam to the front of the room.
- You may use built-in Python functions that do not require import, such as `pow`, `len`, `abs`, `bool`, `int`, `float`, `str`, `max`, and `min`.
- You **may not** use example functions defined on your study guide unless a problem clearly states you can.
- Unless otherwise specified, you are allowed to reference functions defined in previous parts of the same question.
- You may not use `;` to place two statements on the same line.

For questions with **circular bubbles**, you should select exactly *one* choice.

- ☐ You must choose either this option
- ☐ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

- ☐ You could select this choice.
- ☐ You could select this one too!

Preliminaries

You can complete and submit these questions before the exam starts.

(a) What is your full name?

Raymond Tsao

(b) What is your student ID number?

3037860126

(c) What is your @berkeley.edu email address?

r112358@berkeley.edu

1 (4.0 points) What would Python Display?

Assume the following code has been executed.

```
bear = -1
oski = lambda print: print(bear)
bear = -2
```

```
s = "Knock"
```

For each expression below, write the output displayed by the interactive Python interpreter when the expression is evaluated. The output may have multiple lines. If an error occurs, write "Error", but include all output displayed before the error. If evaluation would run forever, write "Forever". To display a function value, write "Function". The interactive interpreter displays the value of a successfully evaluated expression, unless it is None.

(a) (1.0 pt) (3 and 4) - 5

- ☐ -2
☒ -1
☐ 1
☐ 2
☐ Error

3 and 4
4-5 = -1
3
True

(b) (1.0 pt) oski(abs)

- ☐ -2
☐ -1
☐ 1
☒ 2
☐ Function
☐ Error

abs(-2) = 2

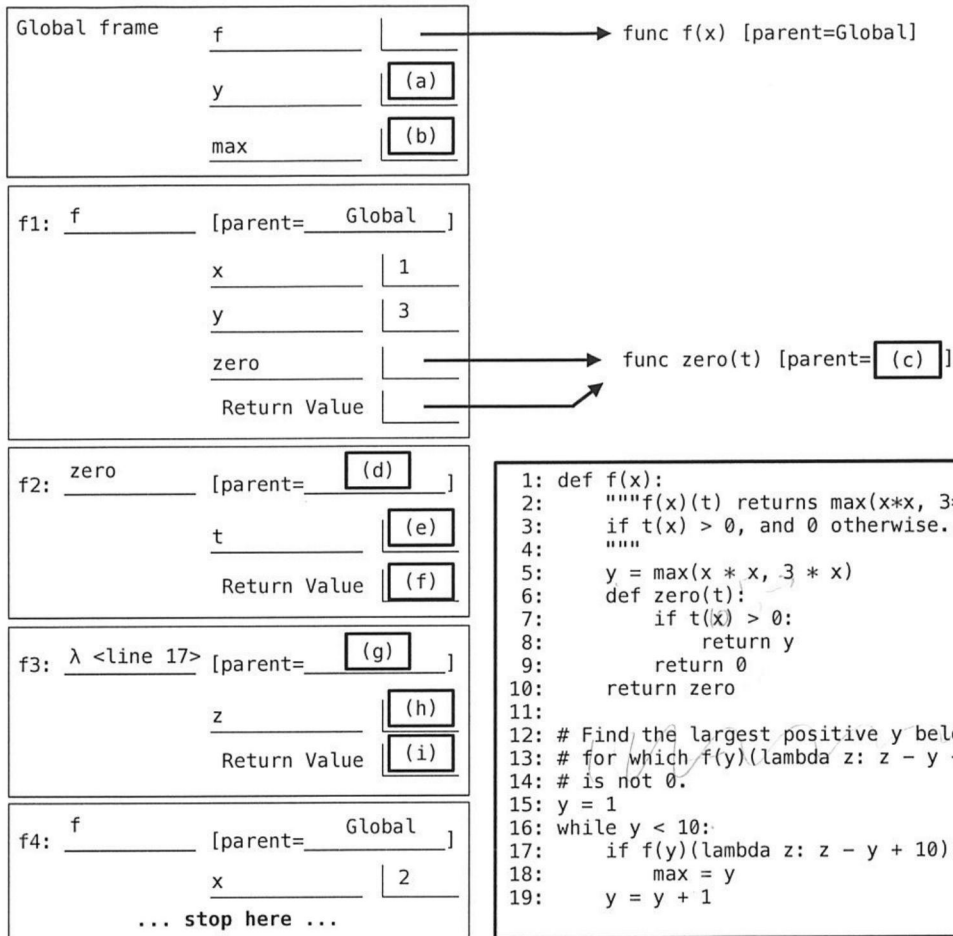
Global
bear L1 L2
oski L → λ(print).
λ
print L → abs

(c) (2.0 pt) print(print(print(s, s) or print("Who's There?")), "Who?")

Knock Knock
Who's There?
None
None Who?

2. (1.0 points) Don't Call Me, I'll Call You

When working on a large project, you get the error 'int' object is not callable in the code below during the second call to f. Complete the environment diagram until the error occurs to identify the problem with the code. If a blank contains an arrow to a function, write the function as it would appear in the diagram.



Global
 f → func f(x) [P=Global]
 y 1 2
 max 1 1

f1. f [P=Global]
 x 1 1
 y 1 3
 zero → zero(t) [P=f1]
 RV 1 1

f2. zero [P=f1]
 t → λ(z) [P=f1]
 RV 1 8

f3. λ [P=f1]
 z 1 1
 RV 1 8

(a) (1.0 pt) Fill in blank (a).

(b) (1.0 pt) Fill in blank (b).

(c) (1.0 pt) Which of these could fill in blank (c)?

- ☐ Global
☐ f
☒ f1

(d) (1.0 pt) Which of these could fill in blank (d)?

☐ Global

☐ f

☒ f1

(e) (2.0 pt) Fill in blank (e).

function $\lambda(z)$ <line 17> [Parent = Global]

(f) (1.0 pt) Fill in blank (f).

3

(g) (1.0 pt) Which of these could fill in blank (g)?

☒ Global

☐ f

☐ zero

☐ f1

☐ f2

(h) (1.0 pt) Fill in blank (h).

1

(i) (1.0 pt) Fill in blank (i).

10

(j) (2.0 pt) Explain in 10 words or less why the 'int' object is not callable error occurred.

In global frame, max has been bounded to an integer.

3/ (10.0 points) It's Perfect

Definitions. A *perfect* number is a positive integer n whose *proper factors* (the factors of n below n) sum to exactly n . A number n is *abundant* if the sum of n 's proper factors is greater than n and *deficient* if that sum is less than n .

(a) (4.0 points)

Implement `classify`, a function that takes an integer n **greater than 1**. It returns the string 'deficient', 'perfect', or 'abundant' that correctly describes n .

```
def classify(n):
    """Return whether n > 1 is 'deficient', 'perfect', or 'abundant'.
```

>>> classify(6) # Proper factors 1, 2 and 3 sum to exactly 6.
'perfect'

>>> classify(24) # Proper factors 1, 2, 3, 4, 6, 8, and 12 sum to 36.
'abundant'

>>> classify(23) # Proper factor 1 sums to 1.
'deficient' *total = 1 < 23*

```
    """
    total, k = 0, 1
    while __<BLANK 1>__: k < n
        if __<BLANK 2>__: n % k == 0
            __<BLANK 3>__ total = total + k 3 + 3
            k = k + 1
    if total == n:
        return 'perfect'
    elif __<BLANK 4>__: total < n
        return 'deficient'
    else:
        return 'abundant'
```

i. (1.0 pt) BLANK 1

- ☒ $k < n$
- ☐ $k \leq n$
- ☐ $k < \text{total}$
- ☐ $k \leq \text{total}$

ii. (1.0 pt) BLANK 2

$n \% k == 0$

iii. (1.0 pt) BLANK 3

$\text{total} = \text{total} + k$

iv. (1.0 pt) BLANK 4

$\text{total} < n$

(b) (6.0 points)

Implement `nearest_perfect`, which takes an integer `n` above 5. It returns the nearest perfect number to `n`. If two perfect numbers are equally close to `n`, return the larger one. A number `a` is nearer to `n` than another number `b` if $\text{abs}(a - n) < \text{abs}(b - n)$. Assume `classify` is implemented correctly.

```
def nearest_perfect(n):
```

```
    """Return the nearest perfect number to n. In a tie, return the larger one.
```

```
    """
    >>> nearest_perfect(8)    # 6 is perfect and 2 away from 8.
    6
```

```
    >>> nearest_perfect(20)  # 28 is perfect and 8 away from 20.
    28
```

```
    >>> nearest_perfect(17)  # Both 6 and 28 are 11 away from 17.
    28
```

```
    >>> nearest_perfect(6)   # 6 is perfect and 0 away from 6.
    6
```

```
    """
```

```
    k = 0
```

```
    while True:
```

```
        if __<BLANK 5>__:
```

```
            __<BLANK 6>__
```

```
        if __<BLANK 7>__:
```

```
            k = -k
```

```
        else:
```

```
            __<BLANK 8>__
```

if classify(n+k) == 'perfect'

return n+k

if classify(n-k) == 'perfect'

k = -k

else

k = k+1

i. (2.0 pt) BLANK 5

classify(n+k) == 'perfect'

ii. (1.0 pt) BLANK 6

☐ `n = n + k`

☐ `n = n + 1`

☐ `n = n - k`

☐ `n = n - 1`

☒ `return n + k`

☐ `return n`

☐ `return k`

iii. (1.0 pt) BLANK 7

classify(n-k) == 'perfect'

iv. (2.0 pt) BLANK 8

k = k+1

4 (4.0 points) Super Powers

Definition. A *function power*, written $f^n(x)$, describes repeated application of a function f to x . $f^2(x) = f(f(x))$, $f^5(x) = f(f(f(f(f(x)))))$, and so on.

- (a) (2.0 pt) Choose **all** correct implementations of `funsquare`, a function that takes a one-argument function `f`. It returns a one-argument function `f2` such that `f2(x)` has the same behavior as `f(f(x))` for all `x`.

```
>>> triple = lambda x: 3 * x
>>> funsquare(triple)(5) # Equivalent to triple(triple(5))
45
```

A: `def funsquare(f):`
`return f(f)` ✓

D: `def funsquare(f):`
`return lambda x: f(f(x))` ✓

B: `def funsquare(f):`
`return lambda: f(f)` ✗

E: `def funsquare(f, x):`
`return f(f(x))` ✓

C: `def funsquare(f, x):`
`def g(x):`
`return f(f(x))`
`return g` ✗

F: `def funsquare(f):`
`def g(x):`
`return f(f(x))`
`return g` ✓

☒ A

☐ B

☐ C

☒ D

☒ E

☒ F

- (b) (1.0 pt) Fill in the blank to assign `quadruple` to a function that takes a number `x` and returns `4x`.

```
>>> quadruple = funsquare(__<BLANK 1>__)
>>> quadruple(3) # 4 * 3
12
```

BLANK 1

`lambda x: 2 * x`

- (c) (1.0 pt) Fill in the blank to assign `funfourth` to a function that takes a one-argument function `f(x)` and returns a one-argument function $f^4(x)$. **You may not use a lambda expression.**

```
>>> funfourth = __<BLANK 2>__
>>> funfourth(triple)(10) # 10 * 3 * 3 * 3 * 3 (triple is defined near the top of the page)
810
```

BLANK 2

`funsquare(funsquare(f))`

`triple(triple(x))`

