INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn’t work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address <EMAILADDRESS>. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

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!

You may start your exam now. Your exam is due at <DEADLINE> Pacific Time. Go to the next page to begin.
Preliminaries

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

(a) What is your full name?

(b) What is your student ID number?

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

(d) Sign (or type) your name to confirm that all work on this exam will be your own. The penalty for academic misconduct on an exam is an F in the course.
1. (8.0 points) What Would Python Display?
Assume the following code has been executed already.

```python
one = 1

def choose(one):
    if big(one):
        print('A')
        if huge(one):
            print('B')
    elif big(one) or huge(one):
        print('C')
    if big(one) or print('D'):
        print('E')
    else:
        print('F')

big = lambda x: x >= one
huge = lambda x: x > one

def which():
    one = 3
    def this():
        return one
    return return this
    one = 4
```

(a) (6.0 pt) Which lines are displayed by the interactive Python interpreter after evaluating `choose(one + one)`? Select all that apply.
- A
- B
- C
- D
- E
- None
- None of the above

(b) (2.0 pt) What is displayed by the interactive Python interpreter after evaluating `which()`?
- 2
- 3
- 4
- 5
- A function
- An error occurs before anything is displayed
2. (8.0 points) Silence of the Lambda

Complete the environment diagram below and then answer the questions that follow. There is one question for each labeled blank in the diagram. The blanks with no labels have no questions associated with them and are not scored. If a blank contains an arrow to a function, write the function as it would appear in the diagram.

(a) (2.0 pt) Fill in blank (a). The line number has been omitted for simplicity.

- [ ] -1
- [ ] 1
- [ ] -10
- [ ] func lambda(n) [parent = Global]
- [ ] func lambda(k) [parent = Global]
- [ ] func lambda(n) [parent = f1]
- [ ] func lambda(k) [parent = f1]

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

- [ ] -1
- [ ] 2
- [ ] 3
- [ ] 10
(c) (1.0 pt) Fill in blank (c).
   - -3
   - -2
   - 20
   - 30
   - None

(d) (1.0 pt) Fill in blank (d).
   - -1
   - 2
   - 3
   - 10

(e) (1.0 pt) Fill in blank (e).
   - Global
   - f1
   - f2
   - f3

(f) (2.0 pt) What is printed by the call to print on line 6?
   - -3
   - -2
   - 20
   - 30
   - None
3. (8.0 points) Nearly Square

Implement `near_square`, which takes positive integer \( n \) and non-negative integer \( k \). It returns the largest integer less than or equal to \( n \) which is the product of two positive integers that differ by \( k \) or less. You may use `solve`, which is provided.

```python
def near_square(n, k):
    """Return the largest integer that is less than or equal to \( n \) and equals \( a \times b \) for some positive integers \( a \) and \( b \) where \( |a - b| \leq k \).
    >>> near_square(125, 0)  # 11 \times 11 = 121 and \( |11 - 11| = 0 \)
    121
    >>> near_square(120, 3)  # 10 \times 12 = 120 and \( |10 - 12| = 2 \)
    120
    >>> near_square(120, 1)  # 10 \times 11 = 110 and \( |10 - 11| = 1 \)
    110
    ""
    while True:
        gap = k
        while _______: # (a)
            x = _______  # (b)
            if _______: # Check if \( x \) is a whole number  # (c)
                return _______  # (d)
                _______  # (e)
        _______  # (f)

def solve(b, c):
    """Returns the largest \( x \) for which \( x \times (x + b) = c \)
    >>> solve(2, 120)  # \( x=10 \) solves \( x \times (x + 2) = 120 \)
    10.0
    >>> solve(2, 121)  # \( x=10.045... \) solves \( x \times (x + 2) = 121 \)
    10.045361017187261
    """
    return (b*b/4 + c) ** 0.5 - b/2
```

(a) \(2.0 \text{ pt} \) Fill in blank \( (a) \). Select all that apply.

- □ gap
- □ gap != 0
- □ gap > 0
- ■ gap >= 0
(b) (2.0 pt) Fill in blank (b).

```python
solve(gap, n)
```

(c) (1.0 pt) Fill in blank (c). The `round` function is demonstrated on the Midterm 1 Study Guide (page 2 righthand side).

- `round(x)`
- `x % 10`
- `x == round(x)`
- `x == x % 10`

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

- `n`
- `k`
- `gap`
- `True`

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

- `n = n - 1`
- `n = k - 1`
- `n = n - k`
- `gap = gap - 1`
- `gap = k - 1`
- `k = k - 1`

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

- `n = n - 1`
- `n = k - 1`
- `n = n - k`
- `gap = gap - 1`
- `gap = k - 1`
- `k = k - 1`
4. (16.0 points) Nice Dice

A dice integer is a positive integer whose digits are all greater than or equal to 1 and all less than or equal to 6.

(a) (8.0 points)

Implement streak, which takes a positive integer n. It returns True if n is a dice integer whose digits are all the same and False otherwise.

```python
def streak(n):
    """Return whether positive n is a dice integer in which all the digits are the same.

    >>> streak(22222)
    True
    >>> streak(4)
    True
    >>> streak(22322)  # 2 and 3 are different digits.
    False
    >>> streak(99999)  # 9 is not allowed in a dice integer.
    False
    ""
    d = n % 10
    if _______:
        (a)
        return False
    while n:
        if _______:
            (b)
            _______  
            (c)
            n = n // 10
    return _______  
(d)
```

i. (2.0 pt) Fill in blank (a).
- d < 1 and d > 6
- *d < 1 or d > 6
- n < 1 and n > 6
- n < 1 or n > 6
- min(n) < 1 and max(n) > 6
- min(n) < 1 or max(n) > 6

The loop checks the remaining digits.
ii. (2.0 pt) Fill in blank (b).

\[ d \neq n \mod 10 \]

iii. (1.0 pt) Fill in blank (c).

- \( d = n \mod 10 \)
- \( d = (n \div 10) \mod 10 \)
- \( \text{result} = \text{False} \)
- \( \text{return False} \)

iv. (1.0 pt) Fill in blank (d).

- \( \text{True} \)
- \( \text{result} \)
- \( d = n \)
- \( d = n \mod 10 \)
v. (2.0 pt) Fill in the blank of `changes`, which takes a dice number `n`. It returns the number of digits of `n` that are different from the digit to their left.

```python
def changes(n):
    """Return the number of adjacent digits that are different in dice number n.
    """
    count = 0
    while n >= 10:
        if _______
            count = count + 1
        n = n // 10
    return count
```

- n != n % 100
- n != n // 10
- n != (n // 10) % 10
- n != (n // 10) % 100
- n % 10 != n % 100
- n % 10 != n // 10
- n % 10 != (n // 10) % 10
- n % 10 != (n // 10) % 100
(b) (2.0 pt) Fill in the blank of \texttt{count\_if}, which takes a one-argument function \textit{f}. It returns a function that takes a positive integer \textit{n} and returns the count of its digits for which \textit{f} returns a true value.

\begin{verbatim}
def count\_if(f):
    """Return a function that takes a positive integer \textit{n} and returns the count of its digits for which \textit{f} returns a true value.
    >>> is\_three = lambda d: d == 3
    >>> count\_threes = count\_if(is\_three) # count\_threes returns the number of threes in \textit{n}
    >>> count\_threes(431663334231) # 3 appears 5 times
    5
    """
    def g(n):
        count = 0
        while n:
            if _______:
                count += 1
            n = n // 10
        return count
    return g
f(n % 10)
\end{verbatim}

(c) (2.0 pt) Fill in the blank of \texttt{count\_at\_least}, which takes a positive integer \textit{k}. It returns a function that takes a positive integer \textit{n} and returns the count of its digits that are greater than or equal to \textit{k}.

\begin{verbatim}
def count\_at\_least(k):
    """Return a function that returns how many of the digits of its argument are at least \textit{k}.
    >>> above\_3 = count\_at\_least(3) # above\_3 returns the # of digits greater than or equal to 3
    >>> above\_3(431663334231)
    9
    """
    return count\_if(_______)
\end{verbatim}

- \texttt{lambda: n \% 10 \textgreater= k}
- \texttt{lambda: n \geq k}
- \texttt{lambda: d \geq k}
- \texttt{lambda: d \geq n \% 10}
- \texttt{lambda d: n \% 10 \geq k}
- \texttt{lambda d: n \geq k}
- \texttt{lambda d: d \geq k}
- \texttt{lambda d: d \geq n \% 10}
- \texttt{lambda d: lambda k: n \% 10 \geq k}
- \texttt{lambda d: lambda k: n \geq k}
- \texttt{lambda d: lambda k: d \geq k}
- \texttt{lambda d: lambda k: d \geq n \% 10}
(d) (4.0 pt) Fill in the blank of times, which takes a non-negative integer \( k \), a positive integer \( n \), and a digit \( d \). It returns True if \( n \) contains digit \( d \) exactly \( k \) times and False otherwise. You may use count_if.

```python
def times(k, n, d):
    """Returns whether \( n \) contains digit \( d \) exactly \( k \) times."

    >>> times(3, 6132344561, 4)  # 4 only appears 2 times
    False
    >>> times(3, 61323445614, 4)  # 4 appears exactly 3 times
    True
    """
    return ______

    k == count_if(lambda i: i == d)(n)
```

(e) (0.0 pt) This A+ question is not worth any points. It can only affect your course grade if you have a high A and might receive an A+. Finish the rest of the exam first!

Fill in the blank to implement streak using count_if. You may not write `else` or `*` or `**`.

The expression is long, so you may use multiple lines to write it.

```python
def streak(n):
    """Return whether positive \( n \) is a dice integer in which all the digits are the same."

    >>> streak(22222)
    True
    >>> streak(4)
    True
    >>> streak(22322)  # 2 and 3 are different digits.
    False
    >>> streak(99999)  # 9 is not allowed in a dice integer.
    False
    """
    return ______

    count_if(lambda d: True)(n) == count_if(lambda d: d > 0 and d < 7 and d == n % 10)(n)
```
No more questions.