CS 61A Fall 2019

Structure and Interpretation of Computer Programs

FINAL

INSTRUCTIONS

- You have 2 hours and 50 minutes to complete the exam.
- The exam is closed book, closed notes, closed computer, closed calculator, except three hand-written 8.5" × 11" crib sheet of your own creation and the provided CS 61A study guides.
- Mark your answers **on the exam itself**. We will *not* grade answers written on scratch paper.

Last name	
First name	
Student ID number	
CalCentral email `_@berkeley.edu`	
TA	
Name of the person to your left	
Name of the person to your right	
All the work on this exam is my own. **(please sign)**	

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

POLICIES & CLARIFICATIONS

- 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 `min`, `max`, `pow`, `len`, `abs`, `sum`, `next`, `iter`, `list`, `sorted`, `reversed`, `tuple`, `map`, `filter`, `zip`, `all`, and `any`.
- You **may not** use example functions defined on your study guide unless a problem clearly states you can.
- For fill-in-the-blank coding problems, we will only grade work written in the provided blanks. You may only write one Python statement per blank line, and it must be indented to the level that the blank is indented.
- Unless otherwise specified, you are allowed to reference functions defined in previous parts of the same question.
- You may use the `Tree` and `Link` classes defined on Page 2 (left column) of the Midterm 2 Study Guide.

1. (11 points) What Would Python Display

For each of the expressions in the table 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 first two rows have been provided as examples.

The interactive interpreter displays the contents of the `repr` string of the value of a successfully evaluated expression, unless it is `None`.

Assume that you have first started `python3` and executed the statements on the left, which cause no errors. Assume that expressions on the right are executed in the order shown. Expressions evaluated by the interpreter have a cumulative effect.

```
def again(x):
    def again(y):
        nonlocal x
        x = x - y
        return x
    return again(x) + x

n = sum([again(z) for z in range(9)])
```

```
s = [[i] for i in range(3)]
s.append(s)
for t in list(s):
    t.extend(range(3, 5))
```

```
class A:
    b = 'one'
    def __init__(self, a):
        f = self.a
        self.a = a[1:]
        self.b = 'two'
        f(a)
    def a(self, b):
        print([b, type(self).b])
    def __repr__(self):
        return str(self.a)

class B(A):
    b = 'three'
    def a(self, c):
        A.a(self, self.b)
```

```
f = lambda g, h: lambda p, q: h(g(q), p)
g = f(lambda half: half // 2, pow)
```

Expression	Interactive Output
`pow(10, 2)`	100
`print(4, 5) + 1`	4 5 Error
(1 pt) `print([1].append(2))`	
(1 pt) `n`	
(1 pt) `len(s)`	
(2 pt) `s[1]`	
(2 pt) `A('four')`	
(2 pt) `B('five')`	
(2 pt) `g(3, 5)`	

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

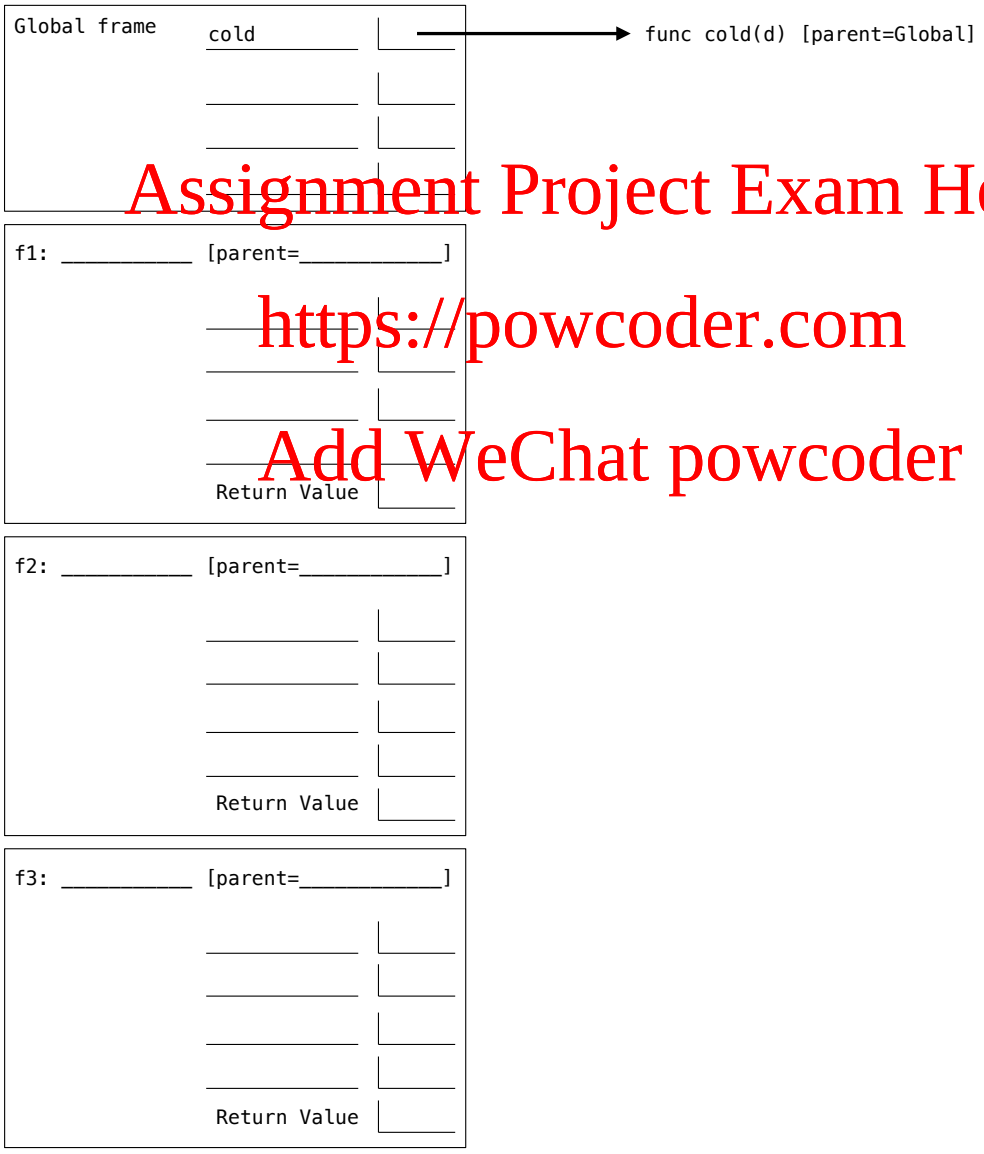
2. (8 points) Protect the Environment

Fill in the environment diagram that results from executing the code on the right until the entire program is finished, an error occurs, or all frames are filled.

Important: You may not need to use all of the spaces or frames. Do not draw frames for calls to built-in functions, such as `len`. A complete answer will:

- Use box-and-pointer notation for lists.
- Add all missing names and parent annotations.
- Add all missing values created or referenced.
- Show the return value for each local frame.

```
1  def cold(d):
2      day = rain[:1]
3      night = lambda: len(day)
4      cold = rain.pop()
5      day = d
6      return night
7
8  rain = [3, 4]
9  d, day = [5], [lambda: d]
10 cold(rain + [day[0](), rain])()
```



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

3. (3 points) The Price is Right

Implement `winner`, which takes a number `price` and returns a function. The function takes a list of numbers `guesses` and returns the largest guess that is less than or equal to the `price`.

Important: Fill each blank with only a single name. You may use built-in functions such as `min`; other built-in functions are listed on the front of the exam.

```
def winner(price):
    """Return a function that takes a list and returns the largest element not above price.

    >>> ipad = winner(499)                  # the iPad actual price is $499
    >>> ipad([500, 600, 200, 1, 350, 299])  # the closest guess that doesn't go over is $350
    350
    """

    return lambda guesses: ________(________(lambda g: ________ <= ________, _____________))
```

4. (4 points) Big Leaf

Implement `tops`, a generator function that takes a `Tree` instance `t`. It yields all leaf labels in `t` for which the path from the root to that leaf is strictly increasing. A sequence is *strictly increasing* if each element is larger than the previous element. A *path* from the root to a leaf is a sequence of labels that includes the root label at the start and the leaf label at the end, along with all labels for nodes in between. The `Tree` class appears on Page 2 (left column) of the Midterm 2 study guide.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Important: You may not modify the attributes of any `Tree` instance or write `else`, `and`, `or`, `[`, or `]`.

```
def tops(t):
    """Yield the leaf labels of Tree t at the end of increasing paths.

    >>> ex = Tree(1, [Tree(2), Tree(5, [Tree(1, [Tree(3)])]), Tree(3, [Tree(4), Tree(1)])])
    >>> list(tops(ex))
    [2, 4]
    >>> list(tops(Tree(1)))
    [1]
    >>> list(tops(Tree(1, [Tree(3, [Tree(2)])])))
    []
    """
    if t.is_leaf():

        yield ____________________________________________________________________________

    else:

        for b in t.branches:

            ____________________________________________________________________________:

                yield from ____________________________________________________________
```

5. (6 points) To-Do Lists

Implement the `TodoList` and `Todo` classes. When a `Todo` is `complete`, it is removed from all the `TodoList` instances to which it was ever added. Track both the number of completed `Todo` instances in each list and overall so that printing a `TodoList` instance matches the behavior of the doctests below. Assume the `complete` method of a `Todo` instance is never invoked more than once.

```
class TodoList:
    """A to-do list that tracks the number of completed items in the list and overall.

    >>> a, b = TodoList(), TodoList()
    >>> a.add(Todo('Laundry'))
    >>> t = Todo('Shopping')
    >>> a.add(t)
    >>> b.add(t)
    >>> print(a)
    Remaining: ['Laundry', 'Shopping'] ; Completed in list: 0 ; Completed overall: 0
    >>> print(b)
    Remaining: ['Shopping'] ; Completed in list: 0 ; Completed overall: 0
    >>> t.complete()
    >>> print(a)
    Remaining: ['Laundry'] ; Completed in list: 1 ; Completed overall: 1
    >>> print(b)
    Remaining: [] ; Completed in list: 1 ; Completed overall: 1
    >>> Todo('Homework').complete()
    >>> print(a)
    Remaining: ['Laundry'] ; Completed in list: 1 ; Completed overall: 2
    """
    def __init__(self):
        self.items, self.complete = [], 0
    def add(self, item):
        self.items.append(item)

        ____________________________________________________________________________________
    def remove(self, item):


        ____________________________________________________________________________ += 1


        self.items.remove(_____________________________________________________________________)

    def __str__(self):
        return ('Remaining: ' + str(_____________________________________________________) +
                ' ; Completed in list: ' + str(self.complete) +


                ' ; Completed overall: ' + str(_____________________________________________))
class Todo:
    done = 0
    def __init__(self, task):
        self.task, self.lists = task, []
    def complete(self):

        ____________________________________________________________________________ += 1
        for t in self.lists:
            t.remove(self)
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

6. (20 points) Palindromes

Definition. A palindrome is a sequence that has the same elements in normal and reverse order.

(a) (3 pt) Implement `pal`, which takes a positive integer `n` and returns a positive integer with the digits of `n` followed by the digits of `n` in reverse order.

Important: You may not write `str`, `repr`, `list`, `tuple`, `[`, or `]`.

```
def pal(n):
    """Return a palindrome starting with n.

    >>> pal(12430)
    1243003421
    """
    m = n

    while m:

        n, m = ____________________________________ , ____________________________________

    return n
```

(b) (4 pt) Implement `contains`, which takes non-negative integers `a` and `b`. It returns whether all of the digits of `a` also appear in order among the digits of `b`.

Important: You may not write `str`, `repr`, `list`, `tuple`, `[`, or `]`.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
def contains(a, b):
    """Return whether the digits of a are contained in the digits of b.

    >>> contains(357, 12345678)
    True
    >>> contains(753, 12345678)
    False
    >>> contains(357, 37)
    False
    """
    if a == b:

        return True

    if ____________________________________ > ____________________________________:

        return False

    if ____________________________________ == ____________________________________:

        return contains( ______________________________ , ______________________________ )

    else:

        return contains( ______________________________ , ______________________________ )
```

(c) (6 pt) Implement `big`, a helper function for `biggest_palindrome`. The `biggest_palindrome` function takes a non-negative integer `n` and returns the largest palindrome integer with an even number of digits that appear among the digits of `n` in order. If there is no even-length palindrome among the digits of `n`, then `biggest_palindrome(n)` returns 0. You may call `pal` and `contains`.

Important: You may not write `str`, `repr`, `list`, `tuple`, `[`, or `]`.

```
def biggest_palindrome(n):
    """Return the largest even-length palindrome in n.

    >>> biggest_palindrome(3425534)
    4554
    >>> biggest_palindrome(126130450234125)
    21300312
    """
    return big(n, 0)

def big(n, k):
    """A helper function for biggest_palindrome."""


    if n == 0:
        return 0

    choices = [big(_____, k), big(_____, _____)]

    if contains(k, _____):
        _____

    return max(choices)
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

(d) (1 pt) Circle the term that fills in the blank: the `is_palindrome` function defined below runs in `____` time in the length of its input.

Constant Logarithmic Linear Quadratic Exponential None of these

```
def is_palindrome(s):
    """Return whether a list of numbers s is a palindrome."""
    return all([s[i] == s[len(s) - i - 1] for i in range(len(s))])
```

Assume that `len` runs in constant time and `all` runs in linear time in the length of its input. Selecting an element of a list by its index requires constant time. Constructing a `range` requires constant time.

(e) (6 pt) Implement `outer`, a helper function for `palinkdrome`. The `palinkdrome` function takes a positive integer `n` and returns a one-argument function that, when called repeatedly `n` times, returns a `Link` containing the sequence of arguments to the repeated calls followed by that sequence in reverse order. The `Link` class appears on Page 2 (left column) of the Midterm 2 study guide.

```
def palinkdrome(n):
    """Return a function that returns a palindrome starting with the args of n repeated calls.

    >>> print(palinkdrome(3)(5)(6)(7))
    <5 6 7 7 6 5>
    >>> print(palinkdrome(1)(4))
    <4 4>
    """
    return outer(Link.empty, n)


def outer(r, n):

    def inner(k):
        s = Link(k, _____)
        if n == 1:
            t = _____
            while s is not Link.empty:
                t, s = Link(_____, _____) , _____
            return t
        else:
            return _____
    return _____
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

7. (10 points) Mull It Over

Uh oh! Someone evaluated `(define * +)`. Now `(* 3 2)` evaluates to 5 instead of 6! Let's fix it.
Important: Answer all questions on this page without calling the built-in multiplication procedure.

(a) (3 pt) Implement `mulxy`, which multiplies integers `x` and `y`. **Hint:** `(- 2)` evaluates to -2.

```
;; multiply x by y (without using the * operator).
;; (mulxy 3 4) -> 12           ; 12 = 3 + 3 + 3 + 3
;; (mulxy (- 3) (- 4)) -> 12   ; 12 = - ( -3 + -3 + -3 + -3 )
(define (mulxy x y)

  (cond ((< y 0) (- ____________________________________________________________ ))

        ((= y 0) 0)

        (else ( ___________________ x (mulxy x _________________________________ )))))
```

(b) (2 pt) Implement `mul-expr`, which takes an expression `e` that contains only calls to `*` and numbers. It returns the normal value of `e` under a Scheme interpreter with an unmodified `*` operator that multiplies. You may call the `mul` procedure defined below.
Important: Fill each blank with only a single symbol!

Assignment Project Exam Help

```
;; Multiply together a list of numbers.
;; (mul '(2 3 4 2)) -> 48
(define (mul s) (reduce mulxy s))
```

https://powcoder.com

```
;; Evaluate an expression with only calls to * and numbers.
;; (mul-expr '(* (* 1 2) (* 3 (* 4 1 1) 2))) -> 48
(define (mul-expr e)
```

Add WeChat powcoder

```
  (if (number? e) e

      (_________________ (_________________ _________________ (_________________ e)))))
```

(c) (5 pt) Implement `*-to-mul`, which takes any expression `e`. It returns an expression like `e`, but with all calls to `*` replaced with calls to `mul`. Note that `*` takes an arbitrary number of arguments, while `mul` always takes exactly one argument: a list of numbers. You should account for this difference.

```
;; Convert all calls to * to calls to mul in expression e.
;; (eval (*-to-mul '(* 1 (+ 2 3) (+ 4 5 (* 6 1))))) -> 75
(define (*-to-mul e)
  (if (not (list? e)) e

    (let ((op ______________ ) (rest ________________________________________________ ))

      (if (equal? op '*)

          (list ____________________________________________________________________ )

          (cons op rest)))))
```

8. (4 points) Live Stream

Implement `up`, which takes an infinite stream `s` with no largest element. (That means for every element there is some larger element later in the stream.) It returns a stream of all elements of `s` that are larger than every previous element in `s`.

```
;; Scale all elements of a stream by k.
(define (scale s k) (cons-stream (* (car s) k) (scale (cdr-stream s) k)))

;; A stream of 1 -2 4 -8 16 -32 64 -128 256 -512 ...
(define twos (cons-stream 1 (scale twos -2)))

;; Return a stream of all elements in s larger than all previous elements.
;; (up twos) -> a stream of 1 4 16 64 256 ...
(define (up s)


    (define (rest t)


        (if (> (car t) (car s))
```

Assignment Project Exam Help

```
            ________________________________________________________________________))
```

https://powcoder.com

```
    (cons-stream ___________________  ________________________________________________))
```

Add WeChat powcoder

9. (3 points) Macro Lens

Implement `partial`, a macro that takes a `call` expression that is missing its last operand. A call to `partial` evaluates to a one-argument procedure that takes a value `y` and returns the result of evaluating `call` extended to include an additional operand `y` at the end.

```
;; A macro that creates a procedure from a partial call expression missing the last operand.
;; (define add-two (partial (+ 1 1))) ->  (lambda (y) (+ 1 1 y))
;; (add-two 3) -> 5 by evaluating (+ 1 1 3)
;;
;; (define eq-5 (partial (equal? (+ 2 3)))) ->  (lambda (y) (equal? (+ 2 3) y))
;; (eq-5 (+ 3 2)) -> #t by evaluating (equal? (+ 2 3) 5)
;;
;; ((partial (append '(1 2))) '(3 4)) -> (1 2 3 4)
(define-macro (partial call)


    `(lambda (y) ____________________________________________________________________ )
```

10. (6 points) Big Game

The `scoring` table has three columns, a `player` column of strings, a `points` column of integers, and a `quarter` column of integers. The `players` table has two columns, a `name` column of strings and a `team` column of strings. Complete the SQL statements below so that they would compute the correct result even if the rows in these tables were different than those shown.

Important: You may write anything in the blanks including keywords such as `WHERE` or `ORDER BY`.

```
CREATE TABLE scoring AS
  SELECT "Donald Stewart" AS player, 7 AS points, 1 AS quarter UNION
  SELECT "Christopher Brown Jr.",    7,           1            UNION
  SELECT "Ryan Sanborn",             3,           2            UNION
  SELECT "Greg Thomas",              3,           2            UNION
  SELECT "Cameron Scarlett",         7,           3            UNION
  SELECT "Nikko Remigio",            7,           4            UNION
  SELECT "Ryan Sanborn",             3,           4            UNION
  SELECT "Chase Garbers",            7,           4;

CREATE TABLE players AS
  SELECT "Ryan Sanborn" AS name,  "Stanford" AS team UNION
  SELECT "Donald Stewart",        "Stanford"         UNION
  SELECT "Cameron Scarlett",      "Stanford"         UNION
  SELECT "Christopher Brown Jr.", "Cal"              UNION
  SELECT "Greg Thomas",           "Cal"              UNION
  SELECT "Nikko Remigio",         "Cal"              UNION
  SELECT "Chase Garbers",         "Cal";
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

(a) (3 pt) Complete the SQL statement below to select a one-column table of quarters in which more than 10 total points were scored.

```
SELECT _____ FROM _____
  GROUP BY quarter _____;
```

1
4

(b) (3 pt) Complete the SQL statement below to select a two-column table of the points scored by each team. Assume that no two players have the same name.

```
SELECT _____ , _____ FROM _____
  WHERE _____;
```

Cal	24
Stanford	20

(*The statement below is not part of any question on the exam.*)

```
SELECT "Thank you, " || name FROM players WHERE SUBSTR(name, 5) = "Chase";
```

11. (0 points) ***Optional*** **Final Thought**

Write a SQL statement that describes your thoughts or feelings about CS 61A.

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder