CS 61A Structure and Interpretation of Computer Programs
Spring 2020

Final Exam

INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn't work, on Google Forms. If either tool stops working, switch to the other one and continue taking the exam. We will merge your solutions together at the end of the exam, taking Google Form submissions in preference to submissions at exam.cs61a.org.

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

Important: Please press "Save" after entering or changing each response throughout the exam. If you have any doubt that the system has saved your responses correctly, email your responses to cs61a@berkeley.edu immediately after the exam ends.

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) Who is your TA? (See cs61a.org/staff.html for pictures.)

1. (14 points) Apply Here

(a) (4 points) max

Implement `max`, which takes a non-empty list of integers and returns the largest.

For example, `(max '(1 4 3 5 2))` evaluates to 5.

The built-in `length` procedure returns the length of a list.

```
(define (max vals) (if (= (length vals) 1) _________
                                            (a)

                       (_________ ((_________ _________))
                           (b)          (c)       (d)

                         (if (> (car vals) giant) (car vals) giant) )))
```

i. (1 pt) What expression completes blank (a)?

ii. (1 pt) Which completes blank (b)?

- ◯ `define`
- ◯ `let`
- ◯ `if`
- ◯ `cond`
- ◯ `begin`

iii. (1 pt) What expression completes blank (c)?

iv. (1 pt) What expression completes blank (d)?

(b) (6 points) partial

Implement `partial`, which takes a procedure `action` and a list `args`. It returns a procedure that takes one argument `final-arg` and returns the result of calling `action` on the values in `args` as well as `final-arg`, in that order.

For example `((partial - '(10 2)) 7)` evaluates to 1, just like `(- 10 2 7)`.

Assume that `action` can be called on `k` arguments, where `k` is one more than the length of `args`.

Hint: The built-in `apply` procedure takes two arguments: a procedure and a list of arguments. It returns the result of calling the procedure on the arguments. For example, `(apply - '(10 2 7))` evaluates to 1, just like `(- 10 2 7)`.

```
(define (partial action args)

  (_________ _________ (apply action (_________ args _________))))
      (a)       (b)                      (c)            (d)
```

i. (1 pt) What completes blank (a)?

ii. (1 pt) Which completes blank (b)?

- ◯ `final-arg`
- ◯ `(final-arg)`
- ◯ `()`
- ◯ `(args final-arg)`

iii. (1 pt) Which symbol completes blank (c)?

Hint: All of these built-in procedures are demonstrated on top of Page 1 of the final study guide.

- ◯ `cons`
- ◯ `list`
- ◯ `append`
- ◯ `car`
- ◯ `cdr`

iv. (3 pt) What expression completes blank (d)?

(c) (4 points) max again

Suppose there were a built-in `maximum` procedure that took one or more numbers as arguments and returned the largest argument. For example, `(maximum 1 4 3 5 2)` would evaluate to 5.

Again, define `max`, which takes a list of numbers and returns its largest element. This time, **you may use only** the symbols `maximum`, `partial`, `apply`, and `list`, along with parentheses, in your implementation. **You may not use any special forms** (such as quotation, lambda, etc.).

Important: `maximum` takes multiple arguments that are numbers, while `max` takes a single argument that is a list of numbers.

For example, `(max '(1 4 3 5 2))` evaluates to 5, just like `(maximum 1 4 3 5 2)`.

```
(define max _________)
               (a)
```

i. (4 pt) You may use only `maximum`, `partial`, `apply`, `list`, and parentheses.

What expression completes blank (a)?

2. (13 points) Contact Tracing

Each row of the `visits` table describes a visit to some establishment by an individual.

- The `who` column indicates the individual's name. Assume everyone has a unique name.
- The `place` column indicates which establishment they visited.
- The `arrive` column indicates the hour they arrived and the `depart` column indicates the hour they departed. Assume all visits begin and end on the hour. All hours are from 1pm to 11pm. Assume `depart` is greater than `arrive`.

```
CREATE TABLE visits AS
SELECT "Oski" AS who, "Bar" AS place, 4 AS arrive, 6 AS depart UNION
SELECT "Oski"       , "Grocery"     , 6          , 8           UNION
SELECT "Oski"       , "Bar"         , 8          , 11          UNION
SELECT "Jane"       , "Grocery"     , 5          , 7           UNION
SELECT "Jane"       , "Restaurant"  , 7          , 8           UNION
SELECT "Jane"       , "Bar"         , 8          , 10          UNION
SELECT "Jack"       , "Restaurant"  , 7          , 9           UNION
SELECT "Jack"       , "Bar"         , 9          , 10;
```

(a) (8 points) contacts

Create the `contacts` table. Each row describes a period of time in which another individual was in the same establishment as Oski. If one individual departs just as another arrives, they do not make contact; they must overlap in time to be included in the `contacts` table.

- The `other` column indicates the name of the individual (not Oski) who came in contact with Oski.
- The `location` column indicates the establishment in which they made contact.
- The `start` column indicates the hour that the contact period began.
- The `stop` column indicates the hour that the contact period ended.

The contents of `contacts` are below. **Any row order is fine.**

```
other | location | start | stop
======|==========|=======|======
Jane  | Grocery  | 6     | 7
Jane  | Bar      | 8     | 10
Jack  | Bar      | 9     | 10
```

Important: Your query should construct `contacts` correctly even if the rows of `visits` were different.

Hint: The `MIN` function computes the smaller of two values when called on two arguments. The `MAX` function computes the larger. These are not aggregate functions when called on two arguments.

```
CREATE TABLE contacts AS

    SELECT b.who                   AS other,
           b.place                 AS location,
           MAX(a.arrive, b.arrive) AS start,
           MIN(a.depart, b.depart) AS stop

    FROM visits AS a, visits AS b

    WHERE a.who = _________ AND _________ AND _________ AND _________;
                    (a)           (b)           (c)           (d)
```

i. (2 pt) What completes blank (a)?

ii. (2 pt) Which expression completes blank (b)?

- ◯ `a.who = b.who`
- ◯ `a.who != b.who`
- ◯ `a.who < b.who`
- ◯ `a.who > b.who`

iii. (2 pt) Which expression completes blank (c)?

- ◯ `stop > start`
- ◯ `start > stop`
- ◯ `stop >= start`
- ◯ `start >= stop`
- ◯ `start = stop`
- ◯ `start != stop`

iv. (2 pt) What expression completes blank (d)?

(b) (5 points) time

Select one row per individual other than Oski. Each row should contain two columns: the `name` of the individual and the total `time` in hours that they spent in contact with Oski.

The result appears below. **Any row order is fine.**

```
name | time
=====|======
Jane | 3
Jack | 1
```

Important: Your query should construct this result correctly even if the rows of `contacts` were different. Assume `contacts` is constructed correctly.

```
SELECT _________ AS name, _________ AS time FROM contacts _________;
          (a)                (b)                              (c)
```

i. (1 pt) What expression completes blank (a)?

- ◯ `who`
- ◯ `name`
- ◯ `other`
- ◯ `"who"`
- ◯ `"name"`
- ◯ `"other"`

ii. (2 pt) What expression completes blank (b)?

iii. (2 pt) What clause completes blank (c)?

3. (7 points) Expression Tree

Definition: A *tree expression* for a `Tree` instance `t` is a string that starts with `t` and contains a Python expression that evaluates to a node label within `t` by using `branches` and `label` attributes and item selection.

For example, if `t = Tree(3, [Tree(4, [Tree(-1)]), Tree(-5)])`, then there are 4 tree expressions for `t`:

- `'t.label'`
- `'t.branches[0].label'`
- `'t.branches[1].label'`
- `'t.branches[0].branches[0].label'`.

The `Tree` class is defined on the Midterm 2 Study Guide.

(a) (4 points) labels

Implement `labels`, which takes a `Tree` instance `t` and returns a list of all tree expressions for `t` in any order.

```
def labels(t):
    """List all tree expressions for tree t.

    >>> t = Tree(3, [Tree(4, [Tree(-1)]), Tree(-5)])
    >>> for e in labels(t):
    ...     print(e)
    t.label
    t.branches[0].label
    t.branches[0].branches[0].label
    t.branches[1].label
    """
    def traverse(t, e):

        result.append(_________)
                        (a)

        for i in range(len(t.branches)):

            traverse(t.branches[i], _________)
                                        (b)
    result = []
    traverse(t, 't')
    return result
```

i. (2 pt) What expression completes blank (a)?

ii. (2 pt) What expression completes blank (b)?

(b) (3 points) smallest

Complete the expression below that evaluates to the node label within `t` that is closest to zero (i.e., has the smallest absolute value). For `t = Tree(3, [Tree(4, [Tree(-1)]), Tree(-5)])`, this would evaluate to `-1`.

Assume `labels` is implemented correctly and the name `t` is bound to a `Tree` instance in the current environment. The built-in `eval` function takes a string and returns the result of evaluating the expression contained in the string in the current environment.

Important: Complete each blank with a single name.

```
_________([eval(_________) for e in labels(t)], key=_________)
   (a)             (b)                                (c)
```

i. (1 pt) What name completes blank (a)?

ii. (1 pt) What name completes blank (b)?

iii. (1 pt) What name completes blank (c)?

No more questions.