CS 1110

Prelim 1 Review
Fall 2023
Exam Info

• **Prelim 1**: Thursday, October 12th at 7:30 pm
  - Last name **A – C** in Ives 305
  - Last name **D – E** in Ives 105
  - Last name **F – Q** in Statler Aud.
  - Last name **R – Z** in Uris G01
  - SDS Students will get an e-mail

• Exceptions ONLY if you filed a conflict

• Grades promised Sunday, October 15th
Studying for the Exam

• Read study guides, review slides online
  ▪ Solution to review posted after review

• Review all labs and assignments
  ▪ Solutions to Assignment 2 are in CMS
  ▪ No solutions to code, but talk to TAs

• Look at exams from past years
  ▪ Exams with solutions on course web page
  ▪ Only look at the fall exams; spring is different
Grading

• We will announce *approximate* letter grades
  ▪ We adjust letter grades based on all exams
  ▪ But no hard guidelines (e.g. mean = grade X)
  ▪ May adjust borderline grades again at final grades
• Use this to determine whether you want to drop
  ▪ **Drop deadline** is next week, October 16\(^{th}\)
  ▪ Will have open office hours on that day to meet
  ▪ Will reach out to students of concern (C or lower)

10/4/23
What is on the Exam?

• Five Questions on the following topics:
  ▪ String slicing functions (A1)
  ▪ Call frames and the call stack (A2)
  ▪ Functions on mutable objects (A3)
  ▪ Testing and debugging (Labs 6 and 10)
  ▪ Short Answer (Terminology)

• + 2 pts for writing your name and net-id
What is on the Exam?

- **Five** Questions on the following topics:
  - String slicing functions (A1)
  - Call frames and the call stack (A2)
  - Functions on mutable objects (A3)
  - Testing and debugging (Labs 6 and 10)
  - Short Answer (Terminology)
- + 2 pts for writing your name and net-id
What is on the Exam?

- **Five** Questions on the following topics:
  - String slicing functions
  - Call frames and the call stack
  - Functions on mutable objects
  - Testing and debugging
  - Short Answer

- + 2 pts for writing your name and net-id

Lists may appear in any of these 5 topics.
What is on the Exam?

• String slicing functions (A1)
  ▪ Will be given a function specification
  ▪ Implement it using string methods, slicing
• Call frames and the call stack (A2)
• Functions on mutable objects (A3)
• Testing and debugging (Labs 6 and 10)
• Short Answer (Terminology)
def make_netid(name, n):

    """Returns: a netid for name with suffix n

    Netid is either two letters and a number (if the student has no middle name) or three letters and a number (if the student has a middle name). Letters in netid are lowercase.

    Example: make_netid('Walker McMillan White', 2) is 'wmw2'
    Example: make_netid('Walker White', 4) is 'ww4'

    Parameter name: the student name
    Precondition: name is a string either with format 'first last'
or 'first middle last'

    Parameter n: the netid suffix
    Precondition: n > 0 is an int."""
Useful String Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.find(s1)</td>
<td>Returns first position of s1 in s; -1 if not there.</td>
</tr>
<tr>
<td>s.rfind(s1)</td>
<td>Returns LAST position of s1 in s; -1 if not there.</td>
</tr>
<tr>
<td>s.lower()</td>
<td>Returns copy of s with all letters lower case</td>
</tr>
<tr>
<td>s.upper()</td>
<td>Returns copy of s with all letters upper case</td>
</tr>
</tbody>
</table>

- We will give you any methods you need
- But you must know how to slice strings!
def make_netid(name, n):
    """Returns: a netid for name with suffix n."""
    name = name.lower() # switch to lower case
    fpos = name.find(' ') # find first space
    first = name[:fpos]
    last = name[fpos+1:]
    mpos = last.find(' ') # see if there is another space
    if mpos == -1:
        return first[0]+last[0]+str(n) # remember, n is not a string
    else:
        middle = last[:mpos]
        last = last[mpos+1:]
        return first[0]+middle[0]+last[0]+str(n)
What is on the Exam?

• String slicing functions (A1)
• Call frames and the call stack (A2)
  - Very similar to A2 (see solution in CMS)
  - May have to draw a full call stack
  - See lectures 4 and 10 (for call stack)
• Functions on mutable objects (A3)
• Testing and debugging (Labs 6 and 10)
• Short Answer (Terminology)
Call Stack Example

- Given functions to right
  - Function `fname()` is not important for problem
  - Use the numbers given
- Execute the call: `lname_first('John Doe')`
- Draw **entire** call stack when helper function `lname` completes line 10
  - Draw nothing else

```python
1. def lname_first(s):
2.     '''Pre: s in the form
3.     'first-name last-name' '''
4.     first = fname(s)
5.     last = lname(s)
6.     return last + ',' + first

7. def lname(s):
8.     '''Pre: same as above'''
9.     end = s.find(' ')
10.    return s[end+1:]
```
Call Stack Example: `lname_first('John Doe')`

1. `def` `lname_first(s):`
2. 
3. """Pre: s in the form 'first-name last-name'""
4. first = `fname(s)`
5. `lname = lname(s)`
6. `return` last + ',' + first

8. `def` `lname(s):`
9. """Pre: same as above""
10. `end = s.find(' ')`
11. `return` s[end+1:]
Call Stack Example: `lname_first('John Doe')`

```
1. def lname_first(s):
   2.     """Pre: s in the form 'first-name last-name' ""
   3.     'first-name last-name' ""
   4.     first = fname(s)
   5.     last = lname(s)
   6.     return last + ',' + first
   7.
   8. def lname(s):
   9.     """Pre: same as above""
10.     end = s.find(' ')
11.     return s[end+1:]
```
**Call Stack Example: lname_first('John Doe')**

1. `def lname_first(s):`
   
2. """Pre: s in the form 'first-name last-name' """

3. `first = name(s)`

4. `last = lname(s)`

5. `return last + ',' + first`

---

No variable `last`. Line 5 is not complete.

Line 10 is **complete**. Counter is next line.
Example with a Mutable Object

1. `def cycle_left(p):`
2. """Cycle coords left"
3. `Pre: p a point"""
4. `temp = p.x`
5. `p.x = p.y`
6. `p.y = p.z`
7. `p.z = temp`

- May get a function on a mutable object
  ```python
  >>> p = Point3(1.0,2.0,3.0)
  >>> cycle_left(p)
  ```
- You are not expected to come up w/ the “folder”
  - Will provide it for you
  - You just track changes
- **Diagram all steps**
Example with a Mutable Object

1. `def cycle_left(p):`
2. """Cycle coords left"
3. `Pre: p a point"""
4. `temp = p.x`
5. `p.x = p.y`
6. `p.y = p.z`
7. `p.z = temp`

```python
>>> p = Point3(1.0, 2.0, 3.0)
>>> cycle_left(p)
```

10/4/23
Example with a Mutable Object

1. ```python
def cycle_left(p):

2. """Cycle coords left

3. Pre: p a point"

4. temp = p.x

5. p.x = p.y

6. p.y = p.z

7. p.z = temp

>>> p = Point3(1.0,2.0,3.0)

>>> cycle_left(p)  # Function Call
```

10/4/23
Example with a Mutable Object

1. def cycle_left(p):

2.     """Cycle coords left"

3.     Pre: p a point"

4.     temp = p.x

5.     p.x = p.y

6.     p.y = p.z

7.     p.z = temp

>>> p = Point3(1.0,2.0,3.0)

>>> cycle_left(p)  Function Call
Example with a Mutable Object

1. `def cycle_left(p):
2.     """Cycle coords left
3.     Pre: p a point"
4.     temp = p.x
5.     p.x = p.y
6.     p.y = p.z
7.     p.z = temp

>>> p = Point3(1.0,2.0,3.0)
>>> cycle_left(p)  # Function Call
Example with a Mutable Object

1. `def cycle_left(p):`
2. """Cycle coords left"
3. `Pre: p a point"""
4. `temp = p.x`
5. `p.x = p.y`
6. `p.y = p.z`
7. `p.z = temp`

```python
>>> p = Point3(1.0, 2.0, 3.0)
```
```python
>>> cycle_left(p)
```

Function Call
Example with a Mutable Object

1. `def cycle_left(p):`
2. """Cycle coords left"
3. `Pre: p a point""
4. `temp = p.x`
5. `p.x = p.y`
6. `p.y = p.z`
7. `p.z = temp`

```python
>>> p = Point3(1.0, 2.0, 3.0)
>>> cycle_left(p)
```

Function Call
Example with a Mutable Object

1. `def cycle_left(p):

2.     """Cycle coords left

3. "Pre: p a point""

4.     temp = p.x

5.     p.x = p.y

6.     p.y = p.z

7.     p.z = temp

>>> p = Point3(1.0,2.0,3.0)

>>> cycle_left(p)  # Function Call
What is on the Exam?

- String slicing functions (A1)
- Call frames and the call stack (A2)
- Functions on mutable objects (A3)
  - Given an object type (e.g. class)
  - Attributes will have invariants
  - Write a function respecting invariants
- Testing and debugging (Labs 6 and 10)
- Short Answer (Terminology)
Example from Assignment 3

- Class: RGB
  - Constructor function: RGB(r,g,b)
  - Remember constructor is just a function that gives us back a mutable object of that type
  - Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>int, within range 0..255</td>
</tr>
<tr>
<td>green</td>
<td>int, within range 0..255</td>
</tr>
<tr>
<td>blue</td>
<td>int, within range 0..255</td>
</tr>
</tbody>
</table>
def lighten(rgb):
    """Lighten each attribute by 10%
    Attributes get lighter when they increase.
    Parameter rgb: the color to lighten
    Precondition: rgb an RGB object"""

    pass # implement me
def lighten(rgb):
    
    """Lighten each attribute by 10%""
    red = rgb.red  # puts red attribute in local var
    red = 1.1*red  # increase by 10%
    red = int(round(red,0))  # convert to closest int
    rgb.red = min(255,red)  # cannot go over 255
    # Do the others in one line
    rgb.green = min(255,int(round(1.1*rgb.green,0))))
    rgb.blue = min(255,int(round(1.1*rgb.blue,0))))
Another Example

• Class: Length
  ▪ Constructor function: Length(ft,in)
  ▪ Remember constructor is just a function that gives us back a mutable object of that type
  ▪ Attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>feet</td>
<td>int, non-negative, = 12 in</td>
</tr>
<tr>
<td>inches</td>
<td>int, within range 0..11</td>
</tr>
</tbody>
</table>
def difference(len1, len2):

    """Returns: Difference between len1 and len2
    Result is returned in inches
    Parameter len1: the first length
    Precondition: len1 is a length object longer than len2
    Parameter len2: the second length
    Precondition: len2 is a length object shorter than len1"""

    pass  # implement me
def difference(len1, len2):
    """Returns: Difference between len1 and len2
    Result is returned in inches
    Parameter len1: the first length
    Parameter len2: the second length
    Precondition: len2 is a length object shorter than len1"""
    feetdif = (len1.feet - len2.feet) * 12
    inchdif = len1.inches - len2.inches  # may be negative
    return feetdif + inchdif
What is on the Exam?

- String slicing functions (A1)
- Call frames and the call stack (A2)
- Functions on mutable objects (A3)
- Testing and debugging (Lab 6 and 10)
  - Coming up with test cases
  - Tracing program flow
  - Understanding assert statements
- Short Answer (Terminology)

10/4/23
def pigify(w):
    
    """**Returns**: copy of w converted to Pig Latin
    
    'y' is a vowel if it is not the first letter
    If word begins with a vowel, append 'hay'
    
    If word starts with 'q', assume followed by 'u';
    move 'qu' to the end, and append 'ay'
    
    If word begins with a consonant, move all
    consonants up to first vowel to end and add 'ay'
    
    **Parameter** w: the word to translate
    
    **Precondition**: w contains only (lowercase) letters"""
Picking Test Cases

```python
def pigify(w):
    """Returns: copy of w converted to Pig Latin""
    ...
```

- Test Cases (Determined by the rules):
  - In: 'are', Out: 'arehay'  (Starts with vowel)
  - In: 'quiet', Out: 'ietquay'  (Starts with qu)
  - In: 'ship', Out: 'ipshay'  (Starts with consonant(s))
  - In: 'bzzz', Out: 'bzzzay'  (All consonants)
  - In: 'yield', Out: 'ieldyay'  (y as consonant)
  - In: 'byline', Out: 'ylinebay'  (y as vowel)

10/4/23
def pigify(w):
    """Returns: copy of w converted to Pig Latin""
    ...

• Test Cases (Determined by the rules):
  ▪ In: 'are', Out: 'arehay'  (Starts with vowel)
  ▪ In: 'quiet', Out: 'ietquay'  (Starts with qu)
  ▪ In: 'ship', Out: 'ipshay'  (Starts with consonant(s))
  ▪ In: 'bzzz', Out: 'bzzzay'  (All consonants)
  ▪ In: 'yield', Out: 'ieldyay'  (y as consonant)
  ▪ In: 'byline', Out: 'ylinebaya'  (y as vowel)
def replace_first(word, a, b):
    """Returns: a copy with FIRST instance of a replaced by b
    Example: replace_first('crane','a','o') returns 'crone'
    Example: replace_first('poll','l','o') returns 'pool'
    Parameter word: The string to copy and replace
    Precondition: word is a string
    Parameter a: The substring to find in word
    Precondition: a is a valid substring of word
    Parameter b: The substring to use in place of a
    Precondition: b is a string"""
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""

    pos = word.rfind(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before + b + after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3
pol

polo

'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
kew

'achkew'

Identify the bug(s) in this function.
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b"""
    pos = word.rfind(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before + b + after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3 Unexpected!
pol
polo
'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
kew
'achkew'
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""
    pos = word.find(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before+b+after
    print(result)
    return result

>>> replace_first('pol', 'l', 'o')
3
pol

polo

'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
kew

'achkew'
def replace_first(word, a, b):
    
    """**Returns**: a copy with
    FIRST a replaced by b""

    pos = word.find(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+1:]
    print(after)
    result = before+b+after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3
pol

polo

'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a

kew

UnExpected!

achkew

'achkew'
Debugging Example

```python
def replace_first(word, a, b):
    """Returns: a copy with FIRST a replaced by b""
    pos = word.find(a)
    print(pos)
    before = word[:pos]
    print(before)
    after = word[pos+len(a):]
    print(after)
    result = before+b+after
    print(result)
    return result

>>> replace_first('poll', 'l', 'o')
3
pol
polo
'polo'

>>> replace_first('askew', 'sk', 'ch')
1
a
kew
'achkew'
```

10/4/23
What is on the Exam?

- String slicing functions (A1)
- Call frames and the call stack (A2)
- Functions on mutable objects (A3)
- Testing and debugging (Labs 6 and 10)
- Short Answer (Terminology)
  - See the study guide
  - Look at the lecture slides
  - Read relevant book chapters

10/4/23
Open to Questions
Good Luck!