Exam Info

• Prelim 1: Tuesday, March 12th
  ▪ BKL 219 – Last names A-B
  ▪ BKL 200 – Last names H-K (Balcony) L-S (Main)
  ▪ GSH G76 – Last names C-G
  ▪ GSH 132 - Last names T-Z

• Exceptions ONLY if you filed a conflict
  ▪ We expect you at time and room assigned
  ▪ We will not have pen, pencil, erasers for you – you should be responsible to be prepared for the exam
Studying for the Exam

• Read study guides, review slides online
  ▪ Review slides will be posted after review

• Review all labs and assignments
  ▪ Solutions to A2 are at top of A2 description
  ▪ No solutions to code, but talk to TAs

• Look at exams from past years
  ▪ Exams with solutions on Canvas
  ▪ Spring exams and Fall exam are different
Grading

• We will announce grades through Gradescope
  ▪ 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 March 19\textsuperscript{th}
  ▪ **Goal**: Have everyone graded by end of Thursday
What is on the Exam?

• Questions on the following topics:
  ▪ String slicing functions
  ▪ Call frames and the call stack
  ▪ Functions on mutable objects
  ▪ Testing and debugging
  ▪ Possible short/multiple choice questions
What is on the Exam?

- Questions on the following topics:
  - String slicing functions
  - Call frames and the call stack
  - Functions on mutable objects
  - Testing and debugging
  - Possible short/multiple choice questions

What about lists?
What is on the Exam?

- Questions on the following topics:
  - String slicing functions
  - Call frames and the call stack
  - Functions on mutable objects
  - Testing and debugging
  - Possible short/multiple choice

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

• Questions on the following topics:
  - String slicing functions
    • Do not use magic numbers for index calculations
    • String slicing $<string>[start:end]$  
  - Call frames and the call stack
  - Functions on mutable objects
  - Testing and debugging
  - Possible short/multiple choice questions
3. [18 points] String Slicing.

For this question, you may find the following functions and methods may be useful:

<table>
<thead>
<tr>
<th>Function or Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>len(s)</td>
<td><strong>Returns</strong>: number of characters in s; it can be 0.</td>
</tr>
<tr>
<td>s.find(s1)</td>
<td><strong>Returns</strong>: index of the first character of the FIRST occurrence of s1 in s (-1 if s1 does not occur in s).</td>
</tr>
<tr>
<td>s.rfind(s1)</td>
<td><strong>Returns</strong>: index of the first character of the LAST occurrence of s1 in s (-1 if s1 does not occur in s).</td>
</tr>
<tr>
<td>s.isalpha()</td>
<td><strong>Returns</strong>: True if s is not empty and its elements are all letters; it returns False otherwise.</td>
</tr>
<tr>
<td>s.isdigit()</td>
<td><strong>Returns</strong>: True if s is not empty and its elements are all numbers; it returns False otherwise.</td>
</tr>
<tr>
<td>s.islower()</td>
<td><strong>Returns</strong>: True if s is not empty and its elements are all lowercase letters; it returns False otherwise.</td>
</tr>
<tr>
<td>s.isupper()</td>
<td><strong>Returns</strong>: True if s is not empty and its elements are all uppercase letters; it returns False otherwise.</td>
</tr>
</tbody>
</table>

Recall that a Cornell netid is a string with either 2 or 3 letters (case does not matter), followed by a number. Use this to implement the function below.

```python
def isnetid(s):
    """Return: True if s is a valid netid; False otherwise
    A valid netid is 2 or 3 letters followed by a number.
    Examples: 'wmm2' is a valid netid, but 'wmm2a' and 'w2' are not.
    Precondition: s is string."
```

def isnetid(s):
    if len(s) < 3:
        return False
    if s[2].isdigit():
        pos = 2
    else:
        pos = 3
    prefix = s[:pos].isalpha()
    suffix = s[pos:].isdigit()
    return prefix and suffix

Purpose:
1) Rule out strings that are shorter than 3 characters (because the shortest netid will have two letters and a single digit)
2) Find the position of the where the numbers “should” start (will be either the second or third position)

Why? Because a valid netid is two or three letters followed by numbers!
def isnetid(s):
    if len(s) < 3:
        return False
    if s[2].isdigit():
        pos = 2
    else:
        pos = 3
    prefix = s[:pos].isalpha()
    suffix = s[pos:].isdigit()
    return prefix and suffix

Purpose:
1) Check that the substring s[:pos] are all letters
2) Check that the substring s[pos:] are all numbers

This is why we made the variable pos - to check the prefix/suffix through substrings
What is on the Exam?

• Questions on the following topics:
  ▪ String slicing functions
  ▪ Call frames and the call stack
    • Do NOT follow the Fall semester style
    • Refer to A2 solutions
  ▪ Functions on mutable objects
  ▪ Testing and debugging
  ▪ Possible short/multiple choice questions
Let’s try to see what the global space, heap space, and call stack would look like!
```python
1  def h(y):
2      y = x[1] - y
3      return y
4
5  def g(z, x):
6      x[0] = x[0] - z
7      return x
8
9  def f(x, y, z):
10     x = g(y, z)
11     return h(x[0])
12
13  x = [2, 3]
14  y = [4, 5]
15  x[1] = f(x, 3, y)
```

### Global Space

- **id1**: `x` and `y` are assigned.
- **id2**: `x` and `z` are assigned.

### Call Stack

- **f**
  - `x` and `y` are assigned.
  - `z` is assigned.
  - `return` is executed.
  - `id2` is returned.

- **g**
  - `x` is assigned.
  - `z` is assigned.
  - `return` is executed.
  - `id2` is returned.

- **h**
  - `x` is assigned.
  - `y` is assigned.
  - `return` is executed.
  - `id2` is returned.

### Heap Space

- **id1**: `0: [2, 3]
  1: [2, 3]

- **id2**: `0: [4, 1]
  1: [4, 1]`
What is on the Exam?

• Questions on the following topics:
  ▪ String slicing functions
  ▪ Call frames and the call stack
  ▪ Functions on mutable objects
    • Given an object type (e.g. class)
    • Attributes will have invariants
    • Write a function respecting invariants
  ▪ Testing and debugging
  ▪ Possible short/multiple choice questions
Class Square

• Square has a few attributes:
  - width
  - height
  - \(x\) – represents the position of the left bottom end of the square
  - \(y\) – represents the position of the left bottom end of the square
move(square1, new_x, new_y)

• Implement a function that will, when given a Square object, will set the x and y attributes of the object to the new values given.

• Straightforward in the sense that all you will need to do is change the x and y attributes; you will assign new_x to square1.x and new_y to square1.y.

```python
def move(square1, new_x, new_y):
    square1.x = new_x
    square1.y = new_y
```
has_collided(s1, s2)

- Implement a function that will check if square1 and square2 “collided”; if the two squares have an overlapping region and returns a bool
- Before heading straight into coding, think about the scenarios where the two square objects will have overlapping regions
- What do we know about each square object?
  - The position of the square’s bottom left corner
  - The width and height of the square
Possible scenarios

\[ s1.x < s2.x \text{ and } s1.y < s2.y \]
\[ s2.x < s1.x + s1.width \]
\[ s2.y < s1.y + s1.height \]

\[ s2.x < s1.x \text{ and } s2.y < s1.y \]
\[ s1.x < s2.x + s2.width \]
\[ s1.y < s2.y + s2.height \]

\[ s1.x < s2.x \text{ and } s2.y < s1.y \]
\[ s2.x < s1.x + s1.width \]
\[ s2.y + s2.height < s1.y + s1.height \]

\[ s2.x < s1.x \text{ and } s1.y < s2.y \]
\[ s1.x < s2.x + s2.width \]
\[ s1.y + s1.height < s2.y + s2.height \]
def has_collided(s1, s2):
    first_scenario = (s1.x < s2.x) and (s1.y < s2.y) and (s2.x < s1.x+s1.width) and (s2.y < s1.y+s1.height)
    second_scenario = (s2.x < s1.x) and (s2.y < s1.y) and (s1.x < s2.x+s2.width) and (s1.y < s2.y+s2.height)
    third_scenario = (s1.x < s2.x) and (s2.y < s1.y) and (s2.x < s1.x+s1.width) and (s2.y+s2.height < s1.y+s1.height)
    fourth_scenario = (s2.x < s1.x) and (s1.y < s2.y) and (s1.x < s2.x+s2.width) and (s1.y+s1.height < s2.y+s2.height)
    return first_scenario or second_scenario or third_scenario or fourth_scenario
What is on the Exam?

- Questions on the following topics:
  - String slicing functions
  - Call frames and the call stack
  - Functions on mutable objects
  - Testing and debugging
    - Constructing test cases
    - Figuring out where the code went wrong
    - Understand assert statements
  - Possible short/multiple choice questions
Recall the function before_space

• The function before_space returned the string before the first space character in a given string s
• Precondition of s was that it contained at least one space character
• How can we come up with distinct test cases?
Coming up with test cases

• Let’s first think about the precondition to see what we know about the string $s$
  ▪ It has at least one space character – this means it can have more than one (adjacent? non-adjacent?)
  ▪ Doesn’t have any conditions on where the space character is within $s$ – the space character can be anywhere in the string (start? middle? end?)
• With this, we can construct distinct test cases with rationales for each one
Constructing the test cases

- ‘ abc’ – single space character at the start
- ’abc ‘ – single space character at the end
- ‘a bc’ – single space character in the mid
- ‘  abc’ – many adj. space characters at the start
- ‘abc  ‘ – many adj. space characters at the end
- ‘a  bc’ – many adj. space characters in mid
- ‘a b c’ – many non-adj. space characters
If we follow through the execution, where would the code go wrong?

There is no function named happy_birthday! So, in the middle of executing line_with_name("Teo") in song(), the code will crash!

```
1. def happy_holiday(holiday):
   2.   print("Happy " + holiday)
3.  
4. def dear():
5.   print("Dear " + name)
6.  
7. def to_you():
8.   print("to " + "you")
9.  
10. def line_with_name(name):
11.    happy_birthday(name)
12.    dear(name)
13.  
14. def basic_line(holiday):
15.    happy_holiday(holiday)
16.    to_you()
17.  
18. def song():
19.   basic_line("Birthday")
20.   basic_line("Birthday")
21.   line_with_name("Teo")
22.   basic_line(200)
23.  
24. song()
```
What is on the Exam?

• Questions on the following topics:
  ▪ String slicing functions
  ▪ Call frames and the call stack
  ▪ Functions on mutable objects
  ▪ Testing and debugging
  ▪ Possible short/multiple choice questions
    • See the study guide
    • Look at the lecture slides
    • Read relevant book chapters
Any More Questions?
Good Luck!