CSE 160
Midterm Practice Solutions

1. Evaluate the following Python expressions:

```
(5 / 2) + 2 * 2
   2    + 2 * 2
   2    +   4
        6

["live", "long", "and", "prosper"][1][1:]
"long"[1:]
"ong"

len({1:"one", 2:"two", 3:"three"}[2])
len("two")
3

float(str(2 + 2) + "5") + 1
float(str(4) + "5") + 1
float("4" + "5") + 1
float("45") + 1
45.0 + 1
46.0

itemgetter(1)(["to", "boldly", "go"])
f(["to", "boldly", "go"])            where f(k) is a function that returns k[1]
"boldly"
```

2. Write a function that sorts a list of numbers by their absolute value, and returns a new sorted list as the result. One possible solution:

```
def sort_abs(items):
    return sorted(items, key=abs)
```

3. Write a function that takes a list as a parameter, and returns a set containing the elements that appear more than once in the list. One possible solution:

```
def duplicates(input_list):
    seen = set()
    result = set()
    for element in input_list:
        if element in seen:
            result.add(element)
        seen.add(element)
    return result
```

4. Write a function that takes a string as an argument, and returns a dictionary that maps each character to its frequency in the given string. One possible solution:

```
def freq(input_string):
    result = {}
    for character in input_string:
        if character not in result:
            result[character] = 0
        result[character] = result[character] + 1
    return result
```

5. Write a function that reverses a list, without using the built-in reverse function. Your function should return the reversed list, and not modify the list passed as a parameter. One possible solution:

```
def reverse_list(original_list):
    result = []
    for i in range(len(original_list) - 1, -1, -1):
        result.append(original_list[i])
    return result
```

Another possible solution:

```
def reverse_list(original_list):
    result = []
    for element in original_list:
        result.insert(0, element)
    return result
```

6. For each of the locations indicated above, draw the environment frame(s) at that moment during execution.

Location A:

```
Global Environment              percent_error
a -> 15.0                       actual -> 15.0
b -> 10.0                       expected -> 10.0
pos_dif -> (function)
percent_error -> (function)
```

Location B:

```
Global Environment              percent_error               pos_dif
a -> 15.0                       actual -> 15.0              y -> 15.0
b -> 10.0                       expected -> 10.0            x -> 10.0
pos_dif -> (function)
percent_error -> (function)
```

Location C:

```
Global Environment              percent_error
a -> 15.0                         actual -> 15.0
b -> 10.0                         expected -> 10.0
pos_dif -> (function)             x -> 5.0
percent_error -> (function)       y -> 10.0
```

For more information, execute the code using the Python Tutor.