Work with your neighbor. (This will be graded for participation only.)

1. Consider the following code snippet (there is other code around this that is not shown). Here, `itemlist` is a Python list while `my_dict` is a dictionary.

```
index = int(input())                          # line 0
infile = open('data.txt')                     # line 1
base = int(infile.read())                     # line 2
value = base + index                          # line 3
itemlist[value//index] = my_dict[base]        # line 4
```

   (a) Name four different exceptions that can give occur in the code fragment shown. In each case, give the line number where the exception can arise. **Note**: `index` is a valid integer. Not all line(s) have errors.
   ANS:

   Line 1: FileNotFoundError

   Line 2: ValueError

   Line 3: No errors possible. (When execution hits line 3, `base` must be an integer and the **Note** above says that `index` is a valid integer.)

   Line 4: ZeroDivisionError, or IndexError, or KeyError

2. Download the program `propagate.py`. Make two copies of it.
   a) Modify the code to catch the exception in `fun2()`. Print out an error message in the `except` body statement.

      In which function does the error occur?
      Which function catches the error?
      What does `fun2()` return if the exception occurs?
      Does the program end when the exception occurs?

      ANS:
      The error occurs in `fun1()`
      `fun2()` catches it and print out an error message
      `fun2()` returns None if the exception occurs
      The program does not stop

b) Modify the code to catch the exception in `main()`. Print out an error message in the `except` body statement.

In which function does the error occur?
Which function catches the error?

ANS:

The error occurs in `fun1()`
`main()` catches it and print out an error message.
The program ends at that point because `main()` has been completely executed.

Final exam review (general programming and stacks)

3. (General Programming) Write a snippet of code (not a full function) which prints out all of the powers of 2, from 1 to `n`. If `n` is **not** a power of two, stop after printing the power of 2 that follows it. (Remember the powers of 2 are 2^0, 2^1, 2^2, … That is, they are 1, 2, 4, …)

   Examples: if `n == 17`, then print 1, 2, 4, 8, 16, 32. If `n == 4`, print 1, 2, 4. The numbers should be printed on separate lines.

   ANS:

```
for i in range(n):
    pow2 = 2**i
    print(pow2)
    if pow2 > n:
        break
```

4. (References) What is an alias in Python? Demonstrate how an alias can occur using a Python list. Write the code that creates an alias and show the corresponding diagram.

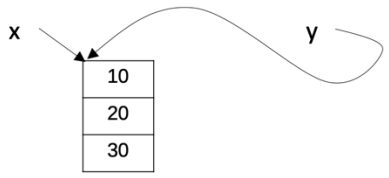
   ANS:

   An alias is the situation where there is more than one variable that references the same object. (I.e., there are two or more names for the same object.)

   A simple example is this:

```
x = [10, 20, 30]
y = x
```

   Both `x` and `y` refer to the same list object. The diagram is below:



5. (Stacks) Write a function `is_balanced(text, lsym, rsym)` that takes three string arguments `text`, `lsym`, and `rsym` and returns `True` if `text` is balanced with respect to the strings `lsym` and `rsym` and `False` otherwise. For example,

   `is_balanced("(a + b) * 3 + 8 * (j + 4)", "(", ")")`

   returns `True` and

   `is_balanced("[4, 6,[8, 2],[3, ]", "[", "]")`

   returns `False`. Use a `Stack` ADT to implement your solution. You may assume that the `Stack` has operations `push(item)`, `pop()`, and `is_empty()` defined.

ANS:

```
def is_balanced(text, lsym, rsym):
    my_stack = Stack()
    bal = True
    for c in text:
        if c == lsym :
            my_stack.push(c)
        elif c == rsym:
            if my_stack.is_empty():
                bal = False
            else:
                my_stack.pop()
    return bal and my_stack.is_empty()
```

<Solutions continued on next page.>

6. Suppose you are given the hash and probe decrement functions below:

```
hash(key) = key % 11
probe(key) = max(1, key / / 11)
```

a) Fill out the columns in the table below for the **Hash value** and **Probe decrement** based on the functions defined above.

ANS:

Key	Hash value	Probe decrement
19	8	1
22	0	2
25	3	2
33	0	3
36	3	3
42	9	3

b) Give the final configuration of a hash table below that results from inserting the keys listed above into an empty table using double hashing. The order of insertion of the keys is: 33, 25, 36, 19, 22, 42.

0	1	2	3	4	5	6	7	8	9	10
33			25			42	19	36	22	

7. We have 42,000 student names to enter as keys into a hash table. If we want a load factor of .70, what should the table size be?

ANS:
The equation for load factor is

$$\lambda = N/M$$

where N is the number of keys and M is the table size.

So,
.70 = 42000 / M

Solving for M gives

M = 42000/.70

So M should be 60,000.