**Q1.** Indicate whether the following statements are true (T) or false (F) by drawing a circle around T or F in each case.

(a) In C, every function has a return statement. T / F

(b) In C, a variable defined in a function can be accessed in another function that is being called from that function. T / F

(c) A C program contains one or more functions but exactly one of those functions must be named as main. T / F

(d) A valid C statements always end in dot (.). T / F

(e) In Python, variable declarations are used to associate specific values to specific identifiers. T / F

(f) A program that keeps running forever and does not stop is an example of a program having a syntax error. T / F

(g) In interpreted programming languages, the source code first needs to be translated to an object code before the execution. T / F

**Q2.** Each code fragment given in the following table will either print something or result in an error message when it is executed. If the code would cause an error, instead write ERROR and briefly explain why an error occurs. Otherwise write down what it will print.

| **Code fragment** | **Output or Cause of Error** |
|---|---|
| `L1 = ["Do", "or", "do", "not"]`<br>`L2 = [L1]`<br>`L2.pop()`<br>`L1.append("There is no try")`<br>`L2.pop()`<br>`print(L2)` | |
| `yoda = {1:'may', 2:'the', 3:'force', 4:'be', 5:'with', 5:'you'}`<br>`print(yoda[-1])` | |
| `r2d2 = ['beep', 'bop', 'boop']`<br>`outL = [word.split('p') for word in r2d2]`<br>`print(outL)` | |
| `myfile = 'A New Hope.txt'`<br>`mylines = myfile.readlines()`<br>`print(mylines[-1])` | |

**Q3.** Suppose you have two files that are each in alphabetic order. For example, the files `ep5.txt` and `ep7.txt` given below:

`sw_ep5.txt`

```
C-3PO
Chewbacca
DarthVader
Emperor
HanSolo
Leia
Lando
Luke
R2-D2
Yoda
```

`sw_ep7.txt`

```
C-3PO
Chewbacca
Finn
HanSolo
KyloRen
Leia
Luke
Poe
R2-D2
Rey
```

Complete the function on the following page according to its docstring description.

```
def find_common_names(file_one, file_two):
    """ (file open for reading, file open for reading) -> list of str

    Precondition: both file_one and file_two contain characters from different
    movies in alphabetic order; each line contains one lowercase word denoting
    the name of a character.

    Return the name of the characters which are common in file_one and file_two
    as a list of strings that is in alphabetic order.

    >>> f1 = open('sw_ep5.txt')
    >>> f2 = open('sw_ep7.txt')
    >>> find_common_names(f1, f2)
    ['C-3PO\n', 'Chewbacca\n', 'HanSolo\n', 'Leia\n', 'Luke\n', 'R2-D2\n']
    """

    common_names = []

    line_one = file_one.readline()
    line_two = file_two.readline()

    while line_one!= "" and line_two!= "":
        if  ____________________________:
            common_names.append(line_one)
            line_one = file_one.readline()
            line_two = file_two.readline()
        elif ____________________________:
             line_one = file_one.readline()
        else:
             line_two = file_two.readline()

    return common_names
```

**Q4.** What does the following program print?

```
def myfunc(a, b, d):
    if (a > b): return [ ]
    elif (a == b): return [ d ]
    else:
        m = (a+b)//2
        return myfunc (a, m-1, d+1) + [d] + myfunc (m+1, b, d+1)
print(myfunc (0, 6, 0))
```

**Q5.** For each of the calls to the following recursive function below, indicate the total number of *'s displayed.

```
def g(n):
   if n==1:
         return 1
   elif n==2:
         print("*")
         return 2
   else:
         print("*")
         return g(n-g(n-1))
```

| **Function call** | **Total number of *'s** |
|---|---|
| g(4) | |
| g(5) | |

**Q6.** What is the output of the following program?

```
#include <stdio.h>

int main(void)
{
      int array[10] = {10, 11, 12, 13, 14, 15, 16, 17, 18, 19};
      int *iptr = (int *)&array;
      void *vptr = &array;

      iptr = iptr + 8;
      vptr = ((char *)vptr) + 8;

      printf("%d\n", *iptr);
      printf("%d\n", *(int *)vptr);

      iptr = (int *) (((double *)iptr) - 2);
      vptr = ((int *)vptr) - 2;

      printf("%d\n", *iptr);
      printf("%d\n", *(int *)vptr);
}
```

**Q7.** Assume *a* and *b* represent two positive integer variables with some values and "^" stands for the binary XOR operator. What does the following C program do?

Hint: Try with some sample *a* and *b* values.

```
a = a ^ b;
b = b ^ a;
a = a ^ b;
```

**Q8.** Please specify the output of the C program below.

```
int main()
{
    const int x = 99;
     int *y;
    y = &x;
    *y = 1907;
    printf("%d\n", x);
    return 0;
}
```

**Q9.** What do the following Python codes print?

| **Code fragment** | **Output or Cause of Error** |
|---|---|
| `a =1`<br>`a, b =a+1, a+1`<br>`print(a, b)` | |
| `a = 5`<br>`b = 3`<br>`a,b = b,a`<br>`print(a, b)` | |
| `print(sum(range(1,5)))` | |
| `a = [1, 2, 2, 3, 3, 3]`<br>`a = list(set(a))`<br>`print(a)` | |
| `n = 1`<br>`print(n++)` | |

**Q10.** What is the output of the following program?

```
def func1():
    global var1
    print(var1)
    var1 = 35

def func2 ():
    global var2
    var2 = (var1 - 3)
    print (var2)

def func3():
    print (var1)
    print (var2)

var1 = "Introduction to Programming"

func1()
func2()
func3()
```

# Sample Questions

**Q1.** Answer the following multiple choice questions by selecting the correct output or the correct explanation for the corresponding code fragments?

-
  ```
  list=[x*x for x in range(10) if x%2==0]
  print(list)
  ```

  a) `[0, 4, 16, 36, 64]`
  b) `[0, 2, 4, 6, 8]`
  c) `[0, 2, 4, 16, 64]`
  d) `[0, 4, 16, 36, 64, 81]`

-
  ```
  a = [1,None,3,{},[],]
  print(len(a))
  ```

  a) `syntax error`
  b) `4`
  c) `5`
  d) `6`

-
  ```
  value1=20.5
  value2=10
  print (value1//value2)
  ```

  a) `2`
  b) `2.25`
  c) `9.0`
  d) `20.25`

-
  ```
  def function(value1, value2):
      pass
  ```

  a) `defines a list and initializes it`
  b) `defines a function, which does nothing`
  c) `defines a function, which passes its parameters through`
  d) `defines an empty class`

-
  ```
  x = True
  y = False
  z = False

  if not x or y:
      print(1)
  elif not x or not y and z:
      print(2)
  elif not x or y or not y and x:
      print(3)
  else:
      print(4)
  ```

  a) `1`
  b) `2`
  c) `3`
  d) `4`

- `print(type([1,2]))`
    a) `<type 'tuple'>`
    b) `<type 'int'>`
    c) `<type 'set'>`
    d) `<type 'list'>`

**Q2.** *Please specify the output of the code given below.*

```
def r2d2(a):
    b = True
    for k in range(len(a)):
        b = b and bb8(a,k)
    return b

def bb8(a,k):
    a[k] = a[k]-1
    return a[k] >= 0

a = [1,2,3,4]
print(r2d2(a))
print(a)
print(r2d2(a))
print(a)
```

**Q3.** What does the following program print?

```
def f():
    a = 2
    print(a)
    return a
def g(b):
    b=10
    return b
def h():
    b=f()
    print(g(b))
h()
```

**Q4.** What does the following code print?

```
def f1():
    global x
    x = 1
    print(x)

def f2():
    x = 2
    print(x)

def f3():
    global x
    print(x)

x = 3
f1()
f2()
f3()
```

**Q5.** *Please specify the output of the code given below.*

```
b = [[9,6],[4,5],[7,7]]
x = b[:2]
x[0] = [1,2]
print(b)
```

**Q6.** Suppose you are given the following function definitions.

```
 def first(x):         def second(x):            def third(x):
     print('S1')           print('S2')               print('S3')
     try:                  try:                      if x < 0:
         second(x)             third(x)                 raise IOError()
     except IOError:       except AssertionError:    elif x > 0:
         print('C1')           print('C2')              raise AssertionError()
     print('E1')           print('E2')               print('E3')
```

a) What is the output of `first(-1)`?

b) What is the output of `first(1)`?

**Q7.** *Suppose you are given the following function definitions.*

```
def female(n):
    if (n == 0):
       return 1
    f = female(n-1)
    return n-male(f)

def male(x):
    if (x == 0):
       return 0
    m = male(x-1)
    return x-female(m)
```

a) While estimating `male(2)`, how many times the functions `female` and `male` are called? What is the resulting value?

i. Number of times `male()` called: ____________

ii. Number of times `female()` called: ____________

iii. Value of `female(3)`: ____________

b) While estimating `female(3)`, how many times the functions `female` and `male` are called? What is the resulting value?

iv. Number of times `male()` called: ____________

v. Number of times `female()` called: ____________

vi. Value of `female(3)`: ____________