Answer: Answers, inline, preceded by red boxes. See exam for full questions and formatting.

Final Examination, Version 4
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

Fall 2025

1. (a) Fill in the code below to produce the output on the right:

```
stops = "Grand Central-42 St,51 St,59 St,68 St-Hunter College"
```

	Python Code:	**Output:**
i.	`print(stops[ ])`	`Gran`

ii.	`counts = {}`
	`for c in stops.upper():`
	`  if :`
	`  counts[c] = counts[c]+1`
	`  else:`
	`  counts[c]=1`
	`print("L appears", counts['L'], "times.")`
		`L appears 3 times.`

iii.	`s_list = stops.` `print(s_list[-1].upper())`	`68 ST-HUNTER COLLEGE`
iv.	`len_s = [ for st in s_list]` `print(len_s[1:3])`	`[5, 5]`
v.	`min_l = min( )` `print("Length of shortest name is", min_l)`	`Length of shortest name is 5`

Answer:

```
boros = "Brooklyn#Bronx#Manhattan#Queens#Staten Island"

print(boros[:5])

counts = {}
for c in boros.lower():
    if c in counts:
        counts[c] = counts[c]+1
    else:
        counts[c] = 1
print("The letter o appears", counts['o'], "times.")
```

```
b_list = boros.split('#')
print(b_list[-1].lower())

shorts = [b[0]+b[-1] for b in b_list]
print(shorts[:3])

b_boros = shorts[:2]
print(len(b_boros),"Boros starting with B")
```

(b) The commands below are **run sequentially**, what is the output after each has run:

```
$ ls -l
-rw-r--r--  1 stjohn  staff    339 Nov  3 16:02 p4.py
-rw-r--r--  1 stjohn  staff    379 Nov 13 15:31 p5.py
-rw-r--r--  1 stjohn  staff    480 Dec  7 14:43 p6V0.py
-rw-r--r--  1 stjohn  staff    736 Dec  3 13:09 p7v0.py
-rw-r--r--  1 stjohn  staff    141 Dec  6 21:01 p9a.cpp
-rw-r--r--  1 stjohn  staff    237 Dec  7 15:14 p9b.cpp
$ pwd
/tmp/v4
```

i.
```
$ mkdir python_progs
$ mv *.py python_progs
$ ls *.cpp
```

Answer:

```
p9a.cpp  p9b.cpp
```

ii.
```
$ echo "Program 9:"
$ ls | grep p9
```

Answer:

```
p9a.cpp  p9b.cpp
```

iii.
```
$ cd python_progs
$ pwd
$ ls -l | grep Nov | wc -l
```

Answer:

```
/tmp/v4/python_progs
2
```

2. (a) Fill in the missing values in the table:

Decimal	Binary	Hexadecimal
2	Answer: 10	2
Answer: 12	1100	C
32	100000	Answer: 20
253	11111101	Answer: FD

(b) Fill in the missing information to make the statements true:

```
import turtle
john = turtle.Turtle()
turtle.colormode(1.0)
john.color(0, Answer: 0,1.0 )

faye = turtle.Turtle()
faye.color("#AAAAAA")
sara = turtle.Turtle()
turtle.colormode(255)
sara.color(200, 0, 200)
zee = turtle.Turtle()
zee.color("# Answer: FF0000 ")

ping = turtle.Turtle()
turtle.colormode(255)
ping.color(200, 0, 0)
```

i. [zee] is red.

ii. [Answer: sara] is purple.

iii. [john] is blue.

iv. [Answer: faye] is gray.

v. [Answer: ping] is bright pink.

(c) Consider the code:

Answer:

```
(i) 1  words = "
(ii)2  while words == "" or len(words) > 10
   3      words = input('Enter a string with 1 to 10 characters: ')
   4  print('You entered:'  words)
```

The answer should include:

- Mark line 1 with a "(i)".
- At end of line 1, should circle the space/parenthesis at the end of the line (where the missing quote should be).
- Mark line 5 with a "(ii)".
- At the end of line 5, should circle the space/parenthesis at the end of the line (where the missing colon should be).

i. **Circle** the code above and mark line with **(i)** that caused this error:

```
words = "
        ^
SyntaxError: unterminated string literal (detected at line 1)
```

Write the code that would fix the error:

Answer:

```
words = ""
```

ii. **Box** the code above and mark line with **(ii)** that caused this error:

```
while words == "" or len(words) > 10
                                    ^
SyntaxError: expected ':'
```

Write the code that would fix the error:

Answer:

```
while words == "" or len(words) > 10:
```

3. (a) What is the value (True/False) of `out`:

i.
```
in1 = True
in2 = False
out = in1 or in2
```

Answer:

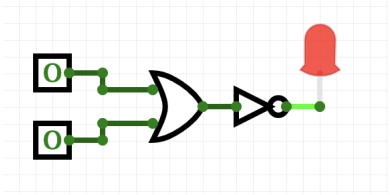
```
out = True
```

ii.
```
in1 = True
in2 = False
out = not in2 or (in2 or not in1)
```

Answer:

```
out = True
```

iii.




```
in1 = True
in2 = False
```

Answer:

```
out = False
```

(b) Fill in the values to yield the output:

i.

```
in1 =
in2 =
```

Answer: `True`

Answer: `False`

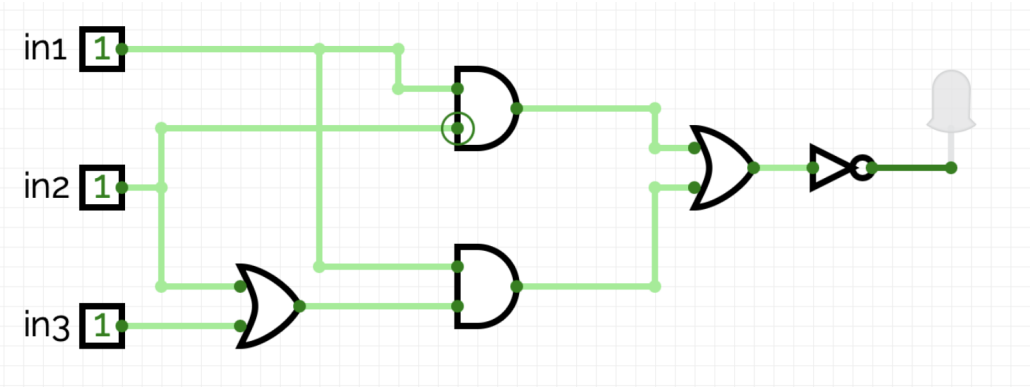
out = `False`

```
out = in1 or (in1 or not in2)
```

(c) Design a circuit that implements the logical expression:

```
not ((in1 and in2) or (in1 and (in2 or in3)))
```

Answer:



4. (a) Draw the output for the function calls:

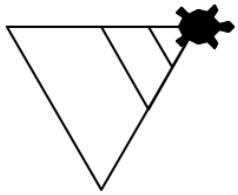
```
1  import turtle
2  tori = turtle.Turtle()
3  tori.shape('turtle')
4
5  def ramble(t, len):
6      if len <= 10:
7          t.stamp()
8      else:
9          for i in range(3):
10             t.right(120)
11             t.forward(len)
12         ramble(t, len//2)
```

i. `ramble(tori,2)`

Answer:

ii. `ramble(tori,80)`

Answer:



(b) What are the formal parameters for `ramble()`:

Answer: `t, len`

(c) If you call `ramble(tori,2)`, which branches of the function are tested (check all that apply):

Answer:

- ✓ The block of code at Line 7.
- ☐ The block of code at Lines 10-11.
- ☐ None of these blocks of code (lines 7, 10-11) are visited from this invocation (call).

(d) If you call `ramble(tori,80)`, which branches of the function are tested (check all that apply):

Answer:

✓ The block of code at Line 7.
✓ The block of code at Lines 10-11.
☐ None of these blocks of code (lines 7, 10-11) are visited from this invocation (call).

5. Design an algorithm that takes a string and returns the most common vowel (that is: 'a','i','e','o', or 'u') in the string. If there are multiple vowels that occurs most often, return the first one alphabetically. Your algorithm, if given the input:

```
"One Fish Two Fish.  -Dr. Seuss"
```

would return **e** since of the three vowels that occur most often ('e','i','o'), 'e' is first alphabetically.

Answer:

Libraries:	none
Input:	a string containing words separated by spaces
Output:	the word that occurs most often

Design Pattern:

Answer: ☐ Accumulator ✓ Max/Min ✓ Finding Duplicates ☐ Searching

Principal Mechanisms (select all that apply):

Answer:

✓ Loop	✓ Conditional (if/else)	☐ Recursion	✓ Indexing/slicing
✓ `input()`	✓ Dictionary	☐ List Comprehension	☐ Regular Expressions

Process (as a concise and precise LIST OF STEPS / pseudocode):

(Assume libraries have already been imported.)

Answer:

(a) Input the string from the user.
(b) Use `.lower()` to convert the string to lower case.
(c) Set up an empty dictionary, `new_dict`.
(d) For char in the string:
(e) Check if the char is a vowel.
(f) Check if the char is in the dictionary.
(g) If it is, increment the count
(h) If it isn't, add word with value 1 to the dictionary.
(i) Find the maximum value in the dictionary and return its key. If there's ties, return the first alphabetically.

Note that this can also be done by using `count(v)` where `v` ranges over all the vowels, and then returning one that occurs most often.

6. Fill in the following functions that are part of a program that draws with turtles:

- `getData()`: gets the color and shape of a turtle and the number of sides of a polygon
- `getTurtle()`: returns a turtle with color and shape
- `drawPolygon()`: draws a polygon with n sides using turtle t

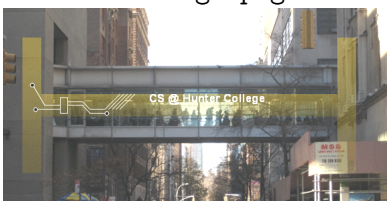
Answer:

```
import turtle
def getData():
      """
      Asks the user for the color and shape of a turtle
      and the number of sides of a polygon.
      Returns the color and shape as strings and the sides as integer.
      """
      color = input('Enter the color of the turtle ')
      shape = input('Enter the shape of the turtle ')
      sides = int(input('Enter the sides of a polygon '))
      return(color, shape, sides)

def getTurtle(color, shape):
      """
      Returns a turtle with color and shape
      """
      t = turtle.Turtle()
      t.color(color)
      t.shape(shape)
      return(t)


def drawPolygon(t, n):
      """
      Draws a polygon with n sides using turtle t
      """
      for i in range(n):
            t.forward(100)
            t.left(360/n)
```

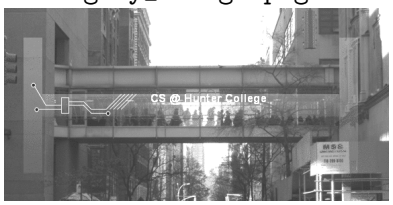
7. Write a **complete Python program** that asks the user for a name of an image .png file and the name of an output file. The program then converts the image to grayscale, by setting the red and blue values for each pixel to its green value. It then saves the grayscale image to the specified output file. A sample run of your program should look like:

```
Enter name of the input file:  csBridge.png
Enter name of the output file:  gray_bridge.png
```

csBridge.png



gray_bridge.png



Answer:

```
#Program 18
import matplotlib.pyplot as plt
import numpy as np

infile = input("Enter the name of an image .png file: ")
outfile = input("Enter the name of an output image file: ")

img = plt.imread(infile)
img[:, :, 0] = img[:, :, 1]
img[:, :, 2] = img[:, :, 1]

plt.imsave(outfile, img)
```

8. (a) Consider the following MIPS program:

```
ADDI $s0, $zero, -1
ADD $s1, $s0, $s0
SUB $s2, $s0, $s1
ADD $s3, $s1, $s1
```

After the program runs, what is the value stored in:

$s0 register	$s1 register	$s2 register	$s3 register
Answer: -1	**Answer:** -2	**Answer:** 1	**Answer:** 2

(b) Consider the MIPS code:

```
1  ADDI $sp, $sp, -7
2  ADDI $t0, $zero, 110
3  ADDI $s2, $zero, 122
4  SETUP: SB $t0, 0($sp)
5  ADDI $sp, $sp, 1
6  ADDI $t0, $t0, 2
7  BEQ $t0, $s2, DONE
8  J SETUP
9  DONE: ADDI $t0, $zero, 0
10 SB $t0, 0($sp)
11 ADDI $sp, $sp, -6
12 ADDI $v0, $zero, 4
```

```
13 ADDI $a0, $sp, 0
14 syscall
```

i) How many characters are printed?	**Answer:** 6
ii) What is the first character printed?	**Answer:** n
iii) What is the whole message printed?	**Answer:** nprtvx
iv) Detail the changes needed to the code to print first half of the message:	**Answer:** Line 1: Change sp to sp - 4. **Answer:** Line 3: Change s2 at 118. **Answer:** Line 11: Subtract 3 from sp.

9. (a) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Rage rage ";
    cout << "against\nthe dying ";
    cout << "of the light."<<endl<<"Dylan";
    cout << "Thomas";
    return 0;
}
```

Answer:

```
Rage rage against
the dying of the light.
Dylan Thomas
```

(b) Fill in the missing code to yield the output:

```
#include <iostream>
using namespace std;
int main()
{
```

Output:

-5	5
0	4
5	3
10	2

Answer:

```
    int myst = -5, quest = 5;

    while ( (myst < 15) && (quest > 0) )
    {
        cout << myst << "\t" << quest << endl;  myst += 5;
        quest--;
    }
    return 0;
}
```

(c) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i <= 5; i++)
    {
        for (int j = 0; j < 5; j++)
            if (i%2 == 0)
                cout << "4";
            else
                cout << "%";
        cout << endl;
    }
    return 0;
}
```

Answer:

```
%%%%%
44444
%%%%%
44444
%%%%%
```

10. (a) Translate the C++ program into a **complete** Python program:

C++ program:

```
#include <iostream>
using namespace std;
int main() {
    float temp = -50.0;
    while ((temp < -10) || (temp > 120))
    {
        cout << "Enter temperature: ";
        cin >> temp;
    }
    cout <<"Temp entered is " <<temp <<".\n";
    return 0;
}
```

Python program:

Answer:

```
temp = -50.0
while temp < -10 or temp > 120:
    temp = float(input('Enter temperature: '))
print('Temp entered is:', temp)
```

(b) Write a **complete C++ program** that asks for a positive whole number, `num`, and prints out the partial products up to `num!` (e.g. 1, 1×2, $1 \times 2 \times 3$,..., $1 \times 2 \times \cdots \times num$).

A sample run of your code:

```
Enter num:  5
1
2
6
24
120
```

Answer:

```
#include <iostream>
using namespace std;

int main()
{
    int num;
    int fact=1;
    cout<< "Enter number: ";
    cin >> num;

    for(int i = 1; i <= num; i++)
    {
        fact = fact * i;
        cout<< fact << endl;
    }
    return(0);
}
```