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

Final Examination, Version 3
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:

```
cuny_man = "Baruch-CCNY-Hunter-John Jay"
```

	Python Code:	**Output:**
i.	`print(cuny_man[ ])`	`Jay`

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

iii.	`c_list = cuny_man.` `print(c_list[-1].upper())`	`JOHN JAY`
iv.	`len_c = [ for c in c_list]` `print(len_c[:3])`	`[6, 4, 6]`
v.	`max_l = max( )` `print("Length of longest is", max_l))`	`8`

Answer:

```
cuny_man = "Baruch-CCNY-Hunter-John Jay"

print(cuny_man[-3:])

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

```
c_list = cuny_man.split('-')
print(c_list[-1].upper())

len_c = [len(c) for c in c_list]
print(len_c[:3])

max_l = max(len_c)
```

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

```
$ ls -l
-rwxr-xr-x  1 stjohn  staff  40354 Nov 29 15:14 a.out
-rw-r--r--  1 stjohn  staff    283 Dec 10 15:14 f25p4aV3.py
-rw-r--r--  1 stjohn  staff     94 Dec 10 16:19 f25V5P2c.py
-rw-r--r--  1 stjohn  staff    218 Nov 29 09:00 p10aV1.cpp
-rw-r--r--  1 stjohn  staff    217 Dec  5 09:06 p10aV2.cpp
-rw-r--r--  1 stjohn  staff    239 Dec  6 17:29 p10aV3.cpp
$ pwd
/tmp/v3
```

i.
```
$ mkdir cprogs
$ mv *.cpp cprogs
$ ls a.*
```

Answer:

```
a.out
$ echo "Cleaning up"
```

ii.
```
$ mkdir pyprogs
$ mv *.py pyprogs
```

Answer:

```
Cleaning up
$ cd cprogs
```

iii.
```
$ pwd
$ ls -l | grep Dec | wc -l
```

Answer:

```
/tmp/v3/cprogs
2
```

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

Decimal	Binary	Hexadecimal
100	Answer: 4	4
Answer: 10	1010	A
35	100011	Answer: 23
255	11111111	Answer: FF

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

```
import turtle
anna = turtle.Turtle()
anna.color("#121212")
isabel = turtle.Turtle()
turtle.colormode(1.0)
```

Answer: `isabel.color(0,1.0,0)`

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

Answer: `daniel.color("#FF0000")`

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

i. `daniel` is red.

ii. Answer: elise is purple.

iii. `isabel` is green.

iv. Answer: anna is gray.

v. Answer: beth is bright pink.

(c) Consider the code:

Answer:

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

(i) (ii)

The answer should include:

- Mark line 2 with a "(i)".
- The space between "len" and "words" is circled.
- Mark line 4 with a "(ii)".
- The space between "entered:" and the comma is boxed.

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

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

Write the code that would fix the error:

Answer:

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

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

```
    print('You entered: , word)
          ^
SyntaxError: unterminated string literal (detected at line 4)
```

Write the code that would fix the error:

Answer:

```
print('You entered:' , word)
```

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

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

Answer:

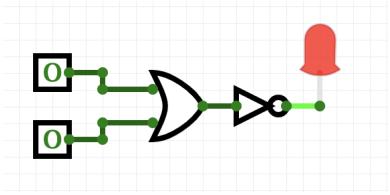
```
out = True
```

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

Answer:

```
out = False
```

iii.




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

Answer:

```
out = True
```

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

i.

```
in1 =  Answer: True
in2 =  Answer: True
```

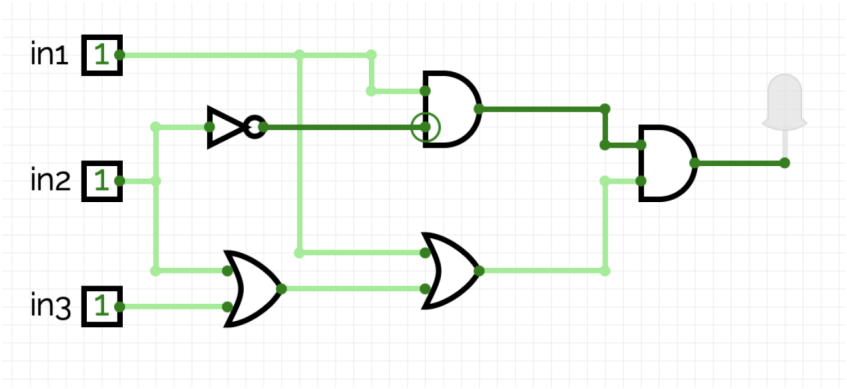
out = `False`

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

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

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

Answer:



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

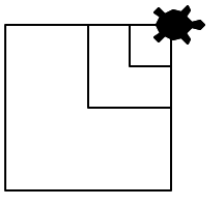
```
1  import turtle
2  tina = turtle.Turtle()
3  tina.shape('turtle')
4
5  def ramble(t, len):
6      if len <= 10:
7          t.stamp()
8      else:
9          for i in range(4):
10             t.right(90)
11             t.forward(len)
12         ramble(t, len//2)
```

i. `ramble(tina,9)`

Answer:

ii. `ramble(tina,80)`

Answer:



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

Answer: `t, len`

(c) If you call `ramble(tina,9)`, 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(tina,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, converts to lower case, and returns the most common character in the string. If there are multiple characters that occur the most often, return all those characters. Your algorithm, if given the input:

```
"Rage rage against the dying --Dylan Thomas"
```

would return **a** since it occurs 6 times, more than other character.

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 in the dictionary.
(f) If it is, increment the count
(g) If it isn't, add word with value 1 to the dictionary.
(h) Find the maximum value in the dictionary and return its key. If there's ties, return all of them.

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:

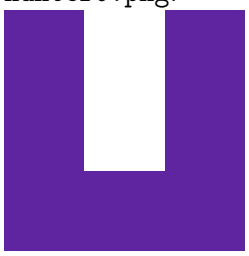
```
import turtle
def getData():
      """
      Asks the user for the color and shape of a turtle
      and the number of sides of a polygon.
      Returns the color and shape as strings and the sides as integer.
      """
      color = input('Enter the color of the turtle ')
      shape = input('Enter 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 creates a 'U' logo for university on a 30x30 grid. Your program should ask the user for amount of red, green, and blue for the color of their logo (specified as values between 0 and 1.0), and the file to save the image.

   A sample run using Hunter College Purple (Pantone 267):

   ```
   Enter amount of red:  0.373
   Enter amount of green:  0.145
   Enter amount of blue:  0.624
   Enter file name:  hunterU.png
   ```

   hunterU.png:

Answer:

```
#Program 21
import numpy as np
```

```
import matplotlib.pyplot as plt

logoImg = np.ones((30, 30, 3))

red = float(input('Enter red: '))
green = float(input('Enter green: '))
blue = float(input('Enter blue: '))

for i,c in zip(range(3),(red,green,blue)):
    print(i,c)
    logoImg[:, :10, i] = c
    logoImg[20:, :, i] = c
    logoImg[:, 20:, i] = c

plt.imshow(logoImg)
plt.show()

fname = input('Enter file name: ')
plt.imsave(fname, logoImg)
```

8. (a) Consider the following MIPS program:

```
ADDI $s0, $zero, -2
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: -2	**Answer:** -4	**Answer:** 2	**Answer:** -8

(b) Consider the MIPS code:

```
1  ADDI $sp, $sp, -6
2  ADDI $t0, $zero, 100
3  ADDI $s2, $zero, 110
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, -5
12 ADDI $v0, $zero, 4
13 ADDI $a0, $sp, 0
14 syscall
```

i) How many characters are printed?	**Answer:** 5
ii) What is the first character printed?	**Answer:** d
iii) What is the whole message printed?	**Answer:** dfhjl
iv) Detail the changes needed to the code to print the half of the message:	**Answer:** Line 1: Change sp to sp - 4. **Answer:** Line 3: Change s2 at 106. **Answer:** Line 11: Subtract 3 from sp.

9. (a) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Had we world\nenough";
    cout << "and time" << endl<<"Andrew";
    cout << "Marvell";
    return 0;
}
```

Answer:

```
Had we world
enough and time
Andrew Marvell
```

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

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

Answer:

```
    int myst = -10, quest = 8;

    while ( (myst < 25) && (quest > 0) )
    {
        cout << myst << "\t" << quest << endl;
        myst += 10;
        quest -=2;
    }
    return 0;
}
```

Output:

-10	8
0	6
10	4
20	2

(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 (j%2 == 0)
                cout << "3";
            else
                cout << "#";
        cout << endl;
    }
    return 0;
}
```

Answer:

```
3#3#3
3#3#3
3#3#3
3#3#3
3#3#3
```

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

C++ program:

```cpp
#include <iostream>
using namespace std;
int main() {
    float gpa = -1.0;
    while ((gpa < 0.0) || (gpa > 4.0))
    {
        cout << "Enter GPA: ";
        cin >> gpa;
    }
    cout << "GPA entered is " << gpa << ".\n";
    return 0;
}
```

Python program:

Answer:

```python
gpa = -1.0
while gpa < 0.0 or gpa > 4.0:
    gpa = float(input('Enter GPA: '))
print('GPA entered is:', gpa)
```

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

A sample run of your code:

```
Enter num:  5
1
3
6
10
15
```

Answer:

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

int main()
{
    int num;
    int sum=0;
    cout<< "Enter number: ";
    cin >> num;

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