Final Examination, Version 1
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

May 2026

1. (a) Assuming the code is run sequentially, what is the output at each step:

i.
```
s = "avram,henriette;dolciani,mary;\
     rees,mina"
a = s[6:11]
print(a.upper())
```

Answer:
```
HENRI
```

ii.
```
names = s.split(';')
print(names[-1])
```

Answer:
```
rees,mina
```

iii.
```
firsts = [n[0] for n in names]
print(firsts)
```

Answer:
```
['a', 'd', 'r']
```

iv.
```
 for n in names:
     w = n.split(',')
     print(w[1],w[0])
```

Answer:
```
henriette avram
mary dolciani
mina rees
```

(b) Consider the following shell commands:

```
$ ls
breads  cakes  cookies  rolls
$ pwd
/Home/bakery
```

Assuming the commands below are run sequentially, what is the output after each has run:

i.
```
$ mkdir desserts
$ ls
```

Answer:
```
breads  cakes   cookies desserts    rolls
```

ii.
```
$ mv c* desserts
$ ls | wc -l
```

Answer:

```
3
```

iii.
```
$ cd desserts
$ echo "Desserts:"
$ ls
```

Answer:

```
Desserts:
cakes   cookies
```

iv.
```
$ touch brownies
$ ls *o*
```

Answer:

```
brownies    cookies
```

2. (a) Check **all that apply**:

Answer:

i. Fill in the code below to change the turtle, `ko`, to be the brightest blue:

`ko.color("#` | 0 | 0 | 0 | 0 | F | F | `")`

ii. Select all the **odd binary numbers**:

☐ 1000 ☐ 1100 ✓ 0111 ☐ 1010 ✓ 1111

iii. Select all the **hexadecimal numbers smaller than 16 decimal**:

✓ A ✓ B ☐ 90 ☐ 99 ☐ FF

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

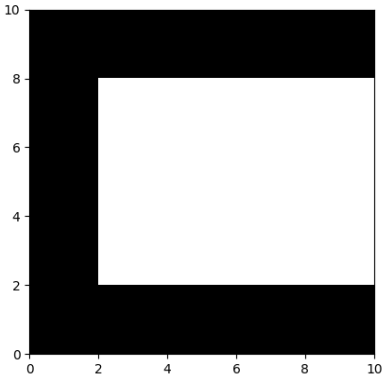
```
import matplotlib.pyplot as plt
import numpy as np
#Set background color to white:

logo = [          ] ( (10,10,3) )
#Draw letter in black:
logo[0:2,:,:] = 0

logo [:, [      ],:]  = 0

logo [ [      ],:,:]  = 0
plt.imshow(logo)
plt.show()
```

Output:



Answer:

```
import matplotlib.pyplot as plt
import numpy as np
#Set background color to white:
logo = np.ones( (10,10,3) )
#Draw letter in black:
logo[0:2,:,:] = 0
logo[:,0:2,:] = 0
logo[-2:,:,:] = 0
plt.imshow(logo, extent= [0,10,0,10])
plt.show()
```

(c) For each error below, give the line number and the code that would fix the error.

```
1  import turtle
2  tess = turtle()
3  for i in range(5)
4      tess.left(90)
5      tess forward(20)
```

i.
```
tess = turtle()
       ^^^^^^^^
TypeError: 'module' object is not callable
```

Answer: **Line Number:** 2 **Correct code:** `tess = turtle.Turtle()`

ii.
```
for i in range(5)
                ^
SyntaxError: expected ':'
```

Answer: **Line Number:** 3 **Correct code:** `for i in range(5):`

iii.
```
tess forward(20)
```

```
      ^^^^^^^
SyntaxError: invalid syntax
```

Answer: **Line Number:** 5 **Correct code:** `tess.forward(20)`

3. (a) What will make the following statement true:

```
in1 and in2
```

Answer:

- ☐ Setting `in1 = False` and `in2 = False`.
- ☐ Setting `in1 = False` and `in2 = True`.
- ☐ All values for `in1` and `in2` make the statement true.
- ☐ Setting `in1 = True` and `in2 = False`.
- ✓ Setting `in1 = True` and `in2 = True`.
- ☐ No values for `in1` and `in2` make the statement true.

(b) What is the value of `out`?

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

Answer:

`out = True`

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

`in1 =` _____

`in2 =` _____

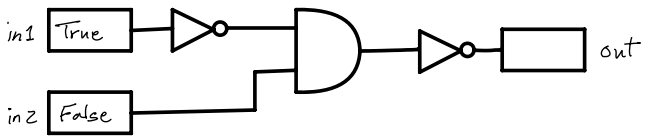
`out =` True

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

Answer:

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

(d) What is the output of this circuit?



Answer:

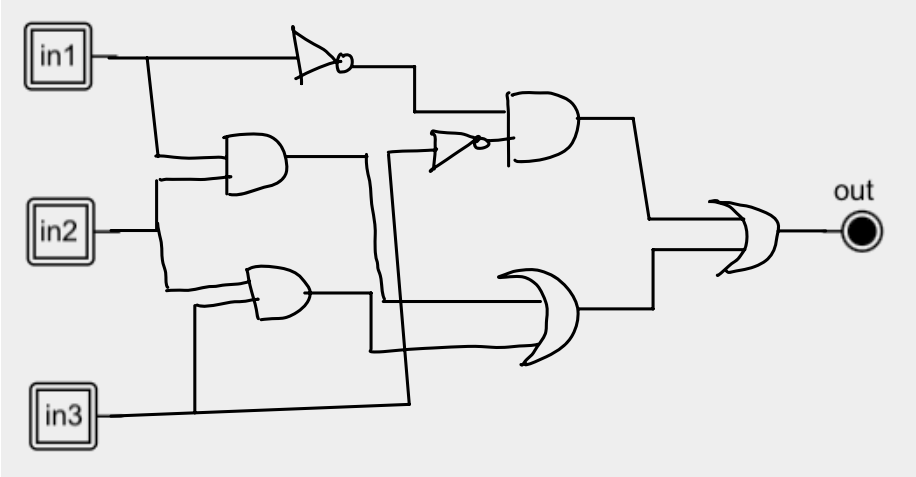
`out = True`

(e) Design a circuit that **exactly implements** the logical expression:

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

Answer:

V1: ((in1 and in2) or (in2 and in3)) or (not in1 and not in3)



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

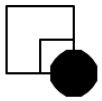
```
1  import turtle
2  taylor = turtle.Turtle()
3  taylor.shape('circle')
4
5  def ramble(tori, num, repeat):
6      if num > 5:
7          for i in range(4):
8              tori.left(90)
9              tori.forward(num)
10         ramble(tori,num//2,repeat)
11     elif repeat:
12         for i in range(num):
13             tori.forward(20)
14             tori.stamp()
15     else:
16         tori.stamp()
```

i. `ramble(taylor,4,True)`

Answer:

ii. `ramble(taylor,30,False)`

Answer:



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

Answer: tori, dist, repeat

(c) If you call `ramble(tyler,4,True)`, which branches of the function are tested (check all that apply): **Answer:**

- ☐ The block of code at Lines 7-10.
- ✓ The block of code at Lines 12-14.
- ☐ The block of code at Line 16.
- ☐ None of these blocks of code (lines 7-10, 12-14, 16) are visited from this invocation (call).

(d) If you call `ramble(tyler,30,False)`, which branches of the function are tested (check all

that apply): **Answer:**

- ✓ The block of code at Lines 7-10.
- ☐ The block of code at Lines 12-14.
- ✓ The block of code at Line 16.
- ☐ None of these blocks of code (lines 7-10, 12-14, 16) 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 character(s) 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 char 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. Complete the following program, based on the payroll dataset below and the comments:

Fiscal Year	Agency Name	Agency Start Date	Work Location Borough	Title Description	Base Salary	Pay Basis	Regular Hours	OT Hours
2018	BOARD OF ELECTION	07/28/2014	MANHATTAN	TEMPORARY CLERK	13.79	per Hour	234.18	75.75
2018	BOARD OF ELECTION	02/28/2016	QUEENS	TEMPORARY CLERK	15	per Hour	1664.55	87
2018	BOARD OF ELECTION	03/13/2016	BRONX	FINANCIAL CLERK	19.79	per Hour	1638.88	66.25
2018	BOARD OF ELECTION	10/02/2017	BRONX	TEMPORARY CLERK	15	per Hour	1195.75	57.5
2018	BOARD OF ELECTION	10/31/2016	BRONX	TEMPORARY CLERK	15	per Hour	1339.38	60.75
2018	BOARD OF ELECTION	06/11/2012	BRONX	TEMPORARY CLERK	15	per Hour	1258.75	58.25

Answer:

```
import pandas as pd

def readDataFrame():
    inFile = input('Enter input file name: ')
    salaries = pd.read_csv(inFile)
    return(salaries)

def alterDataFrame(df):
    newColName = input('Enter the name of the new column: ')
    df[newColName] = (df['Base Salary'] * 1.5) * df['OT Hours']
    return(df, newColName)

def printColumnAverage(df, column):
    avg = df[column].mean()
    print(avg)

def main():
    df = readDataFrame()
    df2, newColName = alterDataFrame(df)
    printColumnAverage(df2, newColName)

if __name__ == '__main__':
    main()
```

7. Write a **complete program** that:

   - Prompts the user to enter a sentence.
   - Splits the input into individual words.
   - Prints every 3rd word from the input, starting with the first word (at index 0).

   A sample run of your program should be:

   ```
   Enter a sentence: The crafty clever Python slithers through code hunting for bugs hiding in nested loops
   ```

   would print:

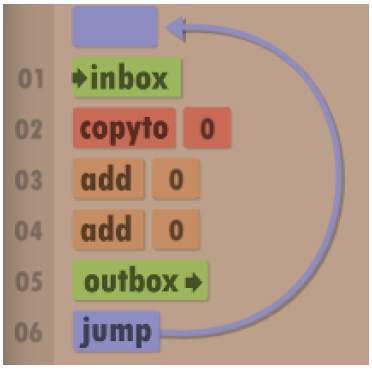
   ```
   The
   Python
   ```

```
code
bugs
nested
```

Answer: sentence=input("Enter a sentence:") words=sentence.split()

for i in range(0,len(words),3): print(words[i])

8. (a) What does the Human Resource Machine (HRM) code output with the following input:



Answer:

9
-15
21
0

(b) Consider the following MIPS program:

```
ADDI $s0, $zero, 1
ADDI $s1, $zero, 6
ADD $s2, $s1, $s1
SUB $s3, $s2, $s0
```

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

$s0 register	$s1 register	$s2 register	$s3 register
Answer: 1	**Answer:** 6	**Answer:** 12	**Answer:** 11

(c) Consider the MIPS code:

```
1  ADDI $sp, $sp, -7
2  ADDI $t0, $zero, 42
3  ADDI $s2, $zero, 6
4  SETUP: SB $t0, 0($sp)
5  ADDI $sp, $sp, 1
6  ADDI $s2, $s2, -1
7  BEQ $s2, $zero, 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
```

Answer:

i) How many characters are printed?	6
ii) What is the first character printed?	*
iii) What is the message printed?	******
iv) List what you need to change to print half of the message:	* line 1: Set $sp to -4. * line 3: Set the counter $s2 to 3. * line 11: Set $sp to -3.

9. (a) Fill in the missing code to yield the output:

```
//Quote by George R.R. Martin
#include <iostream>
using namespace std;
Output:
int main()
{
    cout << "When the snows fall ";
    
    cout << "\nthe lone wolf dies but";
    cout << endl << "the pack survives.\n";
    return 0;
}
```

Output:

```
When the snows fall and
the white winds blow,
the lone wolf dies but
the pack survives.
```

Answer:

```
cout << "and \nthe white winds blow,";
```

(b) What is the output:

```cpp
//More Game of Thrones
#include <iostream>
using namespace std;
int main()
{
    int count = 0;
    while (count < 2) {
       cout <<"If I look back I am lost."<<endl;
       count++;
    }
    cout << "\nNothing burns like the cold\n";
    return 0;
```

Answer:

```
If I look back I am lost.
If I look back I am lost.

Nothing burns like the cold.
```

(c) What is the output:

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

Answer:

```
0@@@@
1####
2@@@@
3####
4@@@@
```

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

Python program:

```
rides = int(input('Enter rides this week: '))
if rides <= 12:
    print('No free rides this week.')
else:
    free_rides = rides - 12
    print(free_rides,'free rides this week.')
```

C++ program:

Answer:

```
#include <iostream>
using namespace std;
int main() {
    int rides;
    cout << "Enter rides this week: ";

    cin >> rides;
    if (rides <= 12)
        cout << "No free rides this week.\n";

    else {
        int free_rides = rides - 12;
        cout << free_rides << " free rides this w
    }
    return 0;
}
```

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

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 prod=1;
    cout<< "Enter number: ";
    cin >> num;
    for(int i = 1; i <= num; i++)
    {
        prod = prod * i;
        cout<< prod << endl;
    }
    return(0);
}
```