Row:	Seat:

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

Fall 2025

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- You may have pens, pencils and one 8 1/2" x 11" reference sheet filled with notes. No other materials are allowed.
- No phones, computers, tablets, calculators, watches, smart glasses, smart pencils, earpods, or other electronic devices are allowed.
- All electronic devices must be turned off and stored in your bag. If you are not able to turn off the Bluetooth/Wifi on your device, put it in your bag at the front of the room.
- **Do not open this exam until instructed to do so.**

Hunter College regards acts of academic dishonesty (e.g., plagiarism, cheating on examinations, obtaining unfair advantage, and falsification of records and official documents) as serious offenses against the values of intellectual honesty. The College is committed to enforcing the CUNY Policy on Academic Integrity and will pursue cases of academic dishonesty according to the Hunter College Academic Integrity Procedures.

I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.									
Name:									
EmpID:									
Email:									
Signature:									

1. (a) What will the following Python code print:

```
rom = {'i' : 1, 'v': 5, 'x' : 10}
s = "v+xvi+xx"
num = s.count('+')+1
print("There are", num, "items.")
rnums = s.split('+')
print("Last number is", rnums[-1])
print("First is", rom[rnums[0]])
sum = 0
for c in rnums[1]:
    sum = sum+rom[c]
    print(sum)
print("Middle is:", sum)
```

Output:

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

```
$ ls
hello.cpp  p1.py  pp_hello.py  setup.sh
$ pwd
/Users/csguest
```

setup.sh

```
echo "Setting up"
mkdir project
cd project
mkdir programs
mkdir data
```

i.
```
$ mv hello.cpp   p1.cpp
$ ls
```

Output:

ii.
```
$ mkdir cprogs
$ mv *.cpp  cprogs
$ ls
```

Output:

iii.
```
$ mkdir pp_5
$ ls | grep pp
```

Output:

iv.
```
$ chmod +x setup.sh
$ ./setup.sh
$ cd project
$ ls
```

Output:

2. (a) Check all that apply:

   i. What color is `tess` after this command? `tess.color("#FF0000")`
   ☐ black ☐ red ☐ white ☐ gray ☐ green

   ii. Select all the **odd** binary numbers:
   ☐ 0000 ☐ 0101 ☐ 0111 ☐ 1010 ☐ 1111

   iii. Select the hexadecimal numbers **smaller than the decimal number 20**:
   ☐ A ☐ 11 ☐ 20 ☐ 23 ☐ FF

   (b) Fill in the code to produce the output on the right:

   i.
```
nums = [ 'a', 'b', 'c', 'd', 'e', 'f']

for i in range(  ,  ,  ):
    print(nums[i], end=" ")
```

Output:

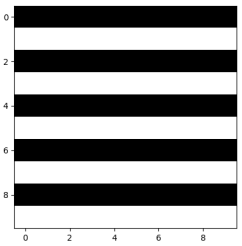
```
f e d
```

```
import numpy as np
import matplotlib.pyplot as plt
img = np.ones( (10,10,3) )
```

   ii. img[, , :] = 0

```
plt.imshow(img)
plt.show()
```

Output:



   (c) Consider the code:

```
1  import turtle
2  tess = turtle.Turtle()
3  for i in range(5)
4      hex_col = input('Enter color (as hex): )
5      tess.color(hex_col)
6      tess.forward(20)
7      tess.stamp()
```

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

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

   Write the code that would fix the error:

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

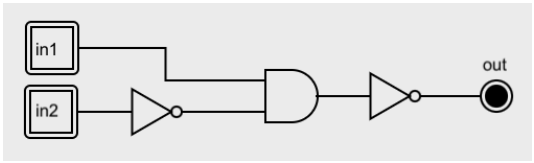
```
    hex_col = input('Enter color (as hex): )
                  ^
SyntaxError: unterminated string literal (detected at line 4)
```

   Write the code that would fix the error:

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

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

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

iii.


out = ____

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

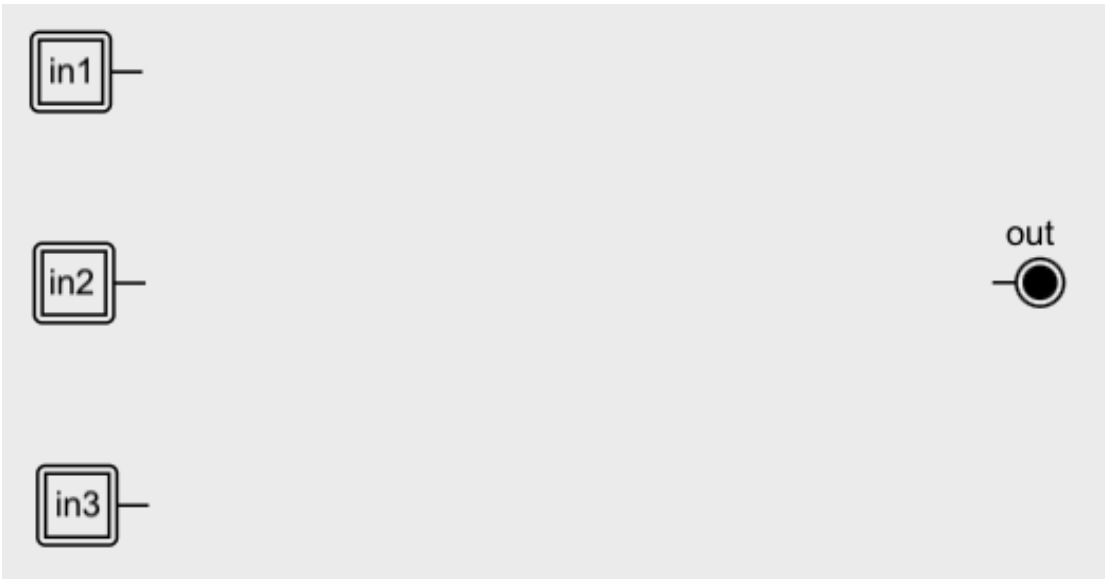
(b) Fill in the values to yield the output:

i.
```
in1 = 
in2 = 

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

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

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

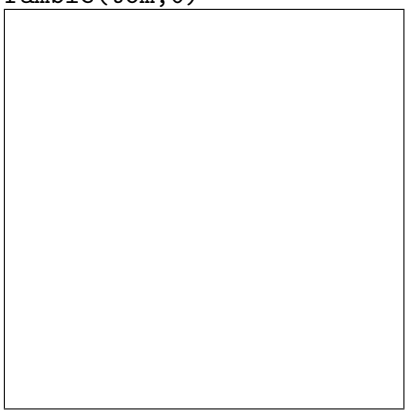


4. (a) Draw the output for the function calls:

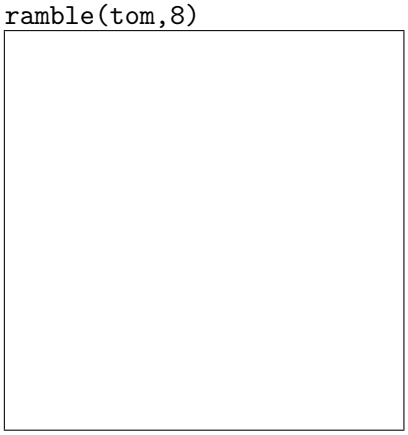
```
import turtle
tom = turtle.Turtle()
tom.shape("circle")

def ramble(t,side):
    if side < 3:
        t.stamp()
    else:
        for i in range(side):
            t.forward(side*10)
            t.left(360/side)
        ramble(t,side-2)
```

i. `ramble(tom,0)`



ii. `ramble(tom,8)`



(b) For the following code:

```
def helper(calvin, reps):
    if reps <= 0:
        return "NA"
    else:
        return (calvin*reps)
```

```
def v1():
    wafa = "a"
    faye = 5
    yun = helper(wafa,faye)
    return yun
```

i. What are the formal parameters for `helper()`:

ii. What are the actual parameters for `helper()` when called in `v1()`:

iii. What value does `v1()` return:

5. Design an algorithm that inputs a string and returns true if, after removing spaces and punctuation, its the same written forward and backwards (e.g. a palindrome). For example, your algorithm should return false for `"Hello Mom!"` since `"hellomom"` and `"momolleh"` are not the same, but would return true for `"A man, a plan, a canal, Panama."` since `"amanaplanacanalpanama"` is the same forward and backwards.

Libraries: (if any)	
Input:	
Output:	

Design Pattern:

☐ Accumulator ☐ Max/Min ☐ Finding Duplicates ☐ Searching

Principal Mechanisms (select all that apply):

☐ 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.)

6. Fill in the missing code below to average regions of an image. For example, if you inputted our favorite image, you would see (left to right):

```
#Fill in libraries needed for storing and displaying images:
```

```
def average(region):
    """
    Returns average of red values, of green values, and blue values
    across the inputted region.
    """
```

```
def setRegion(region, r, g, b):
    """
    Takes a region of an image and red, green, and blue values, r, g, b.
    Sets the region so that all points have
    red values of r, green values of g, and blue values of b.
    """
```

```
def quarter(img2, levels):
     hReg = img2.shape[0]//2**levels
     wReg = img2.shape[1]//2**levels
     for i in range(2**levels):
          for j in range(2**levels):
               r,g,b = average(img2[i*hReg:(i+1)*hReg,j*wReg:(j+1)*wReg])
               setRegion(img2[i*hReg:(i+1)*hReg,j*wReg:(j+1)*wReg],r,g,b)
```

7. Write a **complete Python program** that makes a DataFrame to store addresses and saves the DataFrame in a CSV file. Your program should ask the user for:

   - A list of last names,
   - A list of first names,
   - A list of emails, and
   - The name for the output (CSV) file.

   For example, a sample run of your program:

   ```
   Enter last names: Hunter Raab Kirschner Cantor
   Enter first names: Thomas Jennifer Anne Nancy
   Enter emails: th1870@hunter jr2001@hunter ak2023@hunter nc2024@hunter
   Enter file name:  addr.csv
   ```

   would create a DataFrame:

   ```
           Last     First         emails
   0     Hunter    Thomas  th1870@hunter
   1       Raab  Jennifer  jr2001@hunter
   2  Kirschner      Anne  ak2023@hunter
   3     Cantor     Nancy  nc2024@hunter
   ```

   and save the results to `addr.csv`.

8. (a) Consider the following MIPS program:

```
ADDI $s0, $zero, 3
ADD $s1, $s0, $s0
ADD $s2, $s1, $s0
SUB $s3, $s1, $s2
```

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

$s0 register	$s1 register	$s2 register	$s3 register

(b) Consider the MIPS code:

```
1  ADDI $sp, $sp, -5
2  ADDI $t0, $zero, 50
3  ADDI $s2, $zero, 58
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, -4
12 ADDI $v0, $zero, 4
13 ADDI $a0, $sp, 0
14 syscall
```

i) How many characters are printed?	
ii) What is the first character printed?	
iii) What is the whole message printed?	
iv) Detail the changes needed to the code to print the message in reverse:	

9. (a) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Twinkle, twinkle, little";
    cout << "star,\nhow I wonder";
    cout << "what you are"<<endl<<"Jane";
    cout << "Taylor";
    return 0;
}
```

Output:

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

```
#include <iostream>
using namespace std;
int main()
{
    int myst = -5, quest = 5;
    while ( (myst < 15) && (quest > 0) )
    {
        cout << myst << "\t" << quest << endl;

    }
    return 0;
}
```

Output:

-5	5
0	4
5	3
10	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 < i; j++)
            if (i%2 == 0):
                cout << "1";
            else:
                cout << "@";
        cout << endl;
    }
    return 0;
}
```

Output:

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

Python program:

```
year = 0
while year <= 2026:
    year = int(input('Enter grad year: '))
print("You entered", year)
```

C++ program:

(b) Write a **complete C++ program** that asks for the number of repetitions, print `"Practice makes perfect."` that many times.

A sample run of your code:

```
Enter repetition time:  5
Practice makes perfect.
Practice makes perfect.
Practice makes perfect.
Practice makes perfect.
Practice makes perfect.
```

SCRATCH PAPER

SCRATCH PAPER

CSCI 127 Reference Sheet, Fall 2025

Turtles: Let `t` be a turtle.

Function	Description
`t=Turtle.turtle()`	Creates turtle `t`.
`t.forward(x)`	Moves `t` forward `x` steps.
`t.backward(x)`	Moves `t` backward `x` steps.
`t.left(x)/t.right(x)`	Turns `t` left/right `x` degrees.
`t.penup()/t.pendown()`	Lifts `t`'s pen up/down.
`t.stamp()`	Stamps at `t`'s current location.
`t.goto(x,y)`	Moves `t` to `(x,y)`.

String Methods: Let `s` be a string.

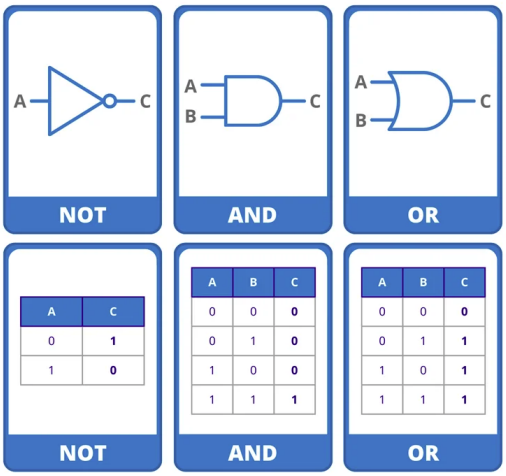
Function	Description
`len(s)`	Returns the length of `s`.
`s.lower()`	Returns `s` as lower case characters.
`s.upper()`	Returns `s` as upper case characters.
`s.count(t)`	Returns count of t in s.
`s.find(t)`	Returns index of t in s (-1 not found).
`s.split(d)`	Splits `s` into list of strings on `d`.
`s.join[lst]`	Joins `lst` into a string, by `s`.
`s[i:j]`	Substring (slice) of s: from i to j-1.
`ord(c)`	Returns Unicode/ASCII of `c`.
`chr(i)`	Returns character of `i`.

Containers: Lists, Ranges & Dictionaries.

Function	Description
`l = []`	Creates an empty list.
`l = [a,b,c]`	List with 3 elements.
`l.append(elt)`	Adds `elt` to end of list.
`l[i]`	Access element at index `i`.
`range(start,stop,step)`	Range object from `start` to `stop-1`, by `step`.
`zip(l1,l2)`	Combines `l1` & `l2` pairwise.
`[x*x for x in l1]`	List of `l1`'s elements squared. (using list comprehension).
`d = {}`	Creates an empty dictionary.
`d = {k1:v2,k2:v2}`	Dictionary of key/value pairs.
`d[k] = v`	Adds `k:v` to dictionary.
`d[k]`	Access value at key `k`.
`k in d`	Checks if key is in dictionary.
`d.keys() / d.values()`	Returns keys/values of `d`.

Functions:

Function	Description
def fname(x,y):	Defines function, `fname`, with
`command1`	(formal) input parameters, `x` and `y`.
`command2...`	Body of function indented.
`return(v)`	Returns value `v`.
`c = fname(a,b)`	Calls/invokes `fname` with (actual) parameters `a` & `b`, returns to `c`.



(from truthtablegen.com)

numpy: Let `np` be the `numpy` package.

Function	Description
`arr_z = np.zeros((10,20,3))`	Sets up array for 10x20 black image.
`arr_1 = np.ones((10,20,z))`	Sets up array for 10x20 white image.
`arr[start:stop:step]`	Slice from `start` to `stop-1` by `step`.
`arr = plt.imread('image.png')`	Read in an image.
`plt.imshow(arr)`	Show `arr` as image.
`plt.show()`	
`plt.imsave('image.png', arr)`	Save an array to file.

Pandas: Let `pd` the Pandas package, `df` be a DataFrame, & `s` a Series.

Function	Description
`pd.read_csv(fn)`	Returns a DataFrame with file `fn`.
`pd.DataFrame(d)`	Returns DataFrame from dictionary `d`.
`df.to_csv(fn)`	Writes `df` to `fn`.
`df[col]`	Returns `col` column as a Series.
`df[[col1,col2]]`	Returns DataFrame with `col1` & `col2`.
`df.columns`	List of column names of `df`.
`df.head(n)/df.tail(n)`	First/last `n` lines of `df`.
`df.plot(x=col)`	Returns a figure with `col` as x-axis.
`fig.savefig(fn)`	Writes `fig` to `fn`.
`s.min()/s.max()/s.mean()`	Returns min/max/average of `s`.
`s.value_counts()`	Counts # times each value occurs.
`df.groupby(col)`	Groups `df` by values in `col`.

Plotly Express: Let `px` be the Plotly Express package.

Function	Description
longitude	Degrees east/west from -180 to 180.
latitude	Degrees north/south from -90 to 90.
px.scatter_geo(df,...)	Returns outline map as fig. Keywords args: `lon,lat,size,hover_name,projection,title.`
px.scatter_map(df,...)	Returns tiled map as fig. Keywords args: `lon,lat,size,hover_name,title,zoom.`
fig.show()	Displays map on browser.
fig.write_html(fn)	Writes `fig` to `fn`.

MIPS: Let `rs`, `rt`, & `rd` be registers.

Function	Description
`ADD rd, rs, rt`	Adds values of rs and rt and stores in rd.
`ADDI rd, rs, imm`	Adds values of rs and imm and stores in rd.
`SUB rd, rs, rt`	Subtracts values of rs and rt and stores in rd.
`BEQ rs, rt, target`	If registers `rs == rt`, jump to `target`.
`JUMP target`	Jump to target.

UNIX:

Function	Description
`ls / ls -l / ls *.py`	Lists files / lists long / lists matching pattern.
`cp x y / mv x y`	Copies/renames file x to file y.
`pwd`	Prints path to current directory.
`mkdir x`	Creates directory called `x`.
`cd ../ / cd /usr/bin`	Changes directory via relative/absolute path.
`echo "message"`	Displays message
`ls|wc -c / ls|grep pat`	Uses pipes to count # of files/match `pat`

C++:

Function	Description
`#include <iostream>`	Includes library with `cin/cout`.
`using namespace std;`	Use standard names w/o `std::`.
`int main() {...}`	Function definition.
`int x;`	Declares variable `x` to be an integer.
`float y;`	Declares variable `y` to be a float.
`cin >> x;`	Reads input into `x`.
`cout << x;`	Prints `x`.
`for (i=0; i<10; i++){...}`	Basic for-loop.
`while (logicalExpression){...}`	Basic while-loop.
`return(v);`	Returns value `v`.

(From wikipedia commons)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]