Row:	Seat:

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

Spring 2025

Exam Rules

- Show all your work. Your grade will be based on the work shown.
- The exam is closed book and closed notes with the exception of an 8 1/2" x 11" piece of paper filled with notes, programs, etc.
- When taking the exam, you may have with you pens and pencils, and your note sheet.
- You may not use a computer, calculator, tablet, phone, earbuds, or other electronic device.
- **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.

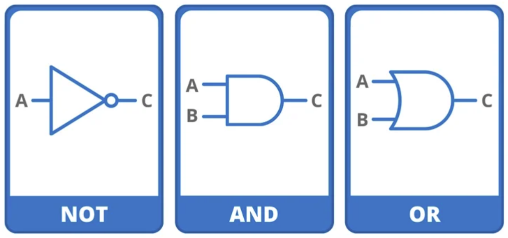
I understand that all cases of academic dishonesty will be reported to the Dean of Students and will result in sanctions.
Name:
EmpID:
Email:
Signature:

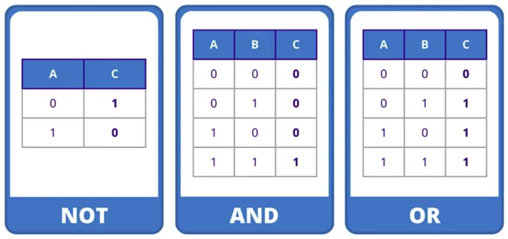
SCRATCH PAPER

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]*

(From wikipedia commons)





A	C
0	1
1	0

NOT

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

OR

(from truthtablegen.com)

Turtles: Let `t` be a turtle.

Function	Description
`t.forward(x)`	Move turtle forward `x` steps.
`t.backward(x)`	Move turtle backward `x` steps.
`t.left(x)`/`t.right(x)`	Turn turtle left/right `x` degrees.
`t.penup()`/`t.pendown()`	Lift turtle's pen up/down.
`t.stamp()`	Stamp at current location.
`t.goto(x,y)`	Move turtle 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.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`.

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

Function	Description
`pd.read_csv(fn)`	Returns a DataFrame with file `fn`.
`pd.to_csv(fn)`	Writes `df` to `fn`.
`pd.DataFrame(d)`	Returns DataFrame built from dictionary `d`.
`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`

SCRATCH PAPER

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

```
mon_s = "January-February-March-April-May-June"
months = mon_s.split('-')
print(len(months),"months")
print("Last month is", months[-1])
short = [mo[:1] for mo in months]
mess = short[-1]
print("Short is:", short)
firsts = {}
for s in short:
    if s in firsts:
        firsts[s] = firsts[s]+1
    else:
        firsts[s] = 1
print("Months with A:", firsts['A'])
print("Months with J:", firsts['J'])
```

Output:

(b) Consider the following shell commands:

```
$ ls
hello.cpp        p1_hello.py        p2_triangle.py
$ pwd
/tmp/final/v1
```

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

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

ii.
```
$ mkdir pyprogs
$ mv *.py pyprogs
$ ls
```
Output:

iii.
```
$ cd pyprogs
$ echo "Current directory:"
$ pwd
```
Output:

iv.
```
$ mkdir old_files
$ cp p1.cpp old_files
$ echo "Count is:"
$ ls | wc -l
```
Output:

2. (a) For each question, **check all that apply:**

   i. What color is `tom` after this command? `tom.color("#AA0000")`?

   ☐ white ☐ green ☐ gray ☐ red ☐ blue

   ii. What is the binary number equivalent to the decimal number 18?

   ☐ 00111 ☐ 01001 ☐ 10010 ☐ 10111 ☐ 11110

   iii. Which of the **binary numbers** below are smaller than the decimal number 9?

   ☐ 10 ☐ 101 ☐ 1010 ☐ 1111 ☐ none

   iv. Select the **smallest** hexadecimal number:

   ☐ AA ☐ 31 ☐ 2C ☐ 1F ☐ FF

   v. Which of the **hexadecimal numbers** below are larger than the decimal number 20?

   ☐ A ☐ F ☐ 19 ☐ 5A ☐ none

(b) After executing the Python code, write the name of the turtle:

```python
import turtle
ellie = turtle.Turtle()
turtle.colormode(1.0)
ellie.color(0.0, 0.0, 1.0)
fatima = turtle.Turtle()
turtle.colormode(255)
fatima.color(255, 0, 0)
guo = turtle.Turtle()
guo.color("#EFEFEF")
hector = turtle.Turtle()
hector.color("#009999")
```

i. which is red:

ii. which is blue-green:

iii. which is blue:

iv. which is gray:

(c) Consider the code:

```python
1   mess == ""
2   while mess == ""
3       mess = input('Enter non-empty string: ')
4   print(mess)
```

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

```
line 1:     mess == ""
            ^^^^
NameError: name 'mess' is not defined
```

Write the code that would fix the error:

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

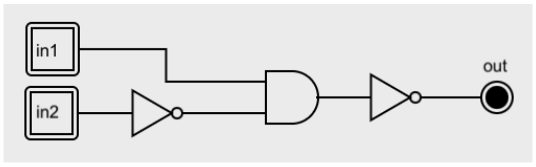
```
line 2: while mess == ""
                        ^
SyntaxError: expected ':'
```

Write the code that would fix the error:

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

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

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

   iii. 

out =

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

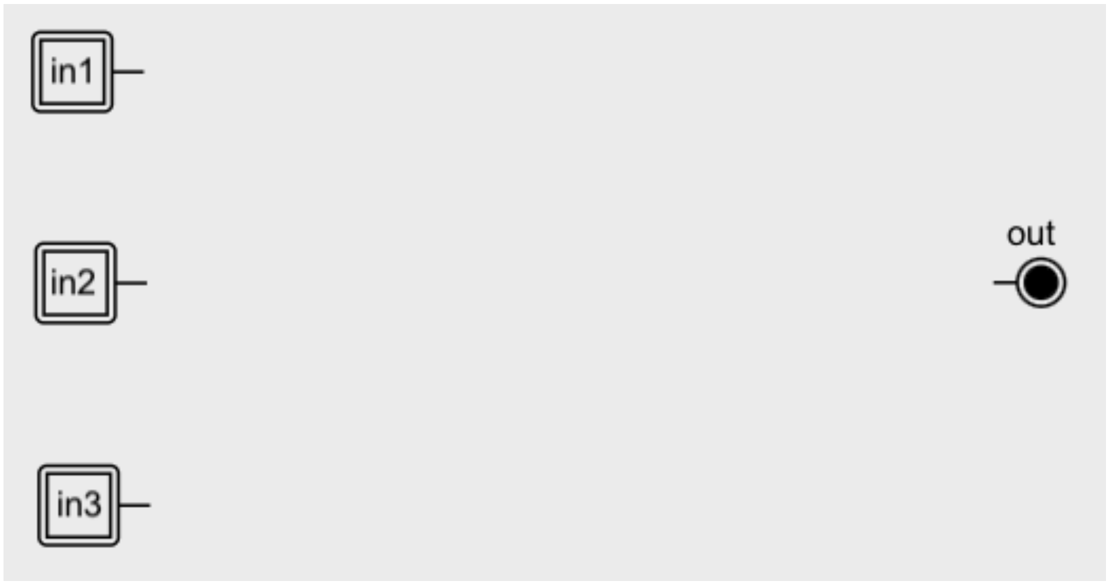
(b) Fill in the values to yield the output:

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

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

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

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



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

```
import turtle
tim = turtle.Turtle()
tim.shape("turtle")

def ramble(t,side):
    if side < 3:
        t.stamp()
    else:
        for i in range(side):
            t.forward(50)
            t.left(360/side)
        ramble(t,side-1)
```

i. `ramble(tim,0)`

ii. `ramble(tim,5)`

(b) For the following code:

```
def v4(antonio, lola):
    if antonio + lola < 10:
        return antonio
    else:
        return -1
```

```
def start():
    jack = 5
    dandan = 20
    kate = v4(jack,dandan)
    return kate
```

i. What are the formal parameters for `v4()`:

ii. What are the formal parameters for `start()`:

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

5. Write a function `unique_visitors()` that takes a list of 8-digit strings and returns the number of unique strings that occur. For example:

```
ids = ['12345678','11223344','12312323','12345678']
unique_visitors(ids)
```

would return `3` since there are 4 entries but the first and fourth entries are duplicates of each other.

Libraries:	
Input:	
Output:	

Design Pattern:

☐ Accumulator ☐ Max/Min ☐ Finding Duplicates ☐ Searching

Principal Mechanisms (select all that apply):

☐ Single Loop ☐ Nested Loop ☐ Conditional (if/else) ☐ Recursion
☐ Indexing/slicing ☐ 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 Python program that will:

   - prompt the user for the name of a CSV file,
   - prompt the user for the name of a column in that CSV file,
   - print out the maximum value of the column,
   - print out the average value of the column, and
   - displays a plot of the column entered (with `"Year"` as the x-axis).

```
#Import the libraries for data frames and displaying images as pd and plt:

#Prompt user for file name:
file_name =

#Prompt user for column name:
col =

#Read in the CSV file to a DataFrame:
df =

#Compute maximum value of the column:

print("Maximum of column", col, "is", M)

#Compute average value of the column:

print("Average of column", col, "is", ave)

#Display a plot of "Year" vs. column entered by user:

```

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

   - asks the user for the name of a `.png` (image) file and
   - prints the number of pixels that are very purple (the fraction of red and the fraction of blue are both above 0.75 and the fraction of green is below 0.25).

8. (a) Consider the following MIPS program:

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

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

$s1 register	$s2 register	$s3 register

(b) Consider the MIPS code:

```
ADDI $sp, $sp, -6
ADDI $t0, $zero, 65
ADDI $s2, $zero, 75
SETUP: SB $t0, 0($sp)
ADDI $sp, $sp, 1
ADDI $t0, $t0, 2
BEQ $t0, $s2, DONE
J SETUP
DONE: ADDI $t0, $zero, 0
SB $t0, 0($sp)
ADDI $sp, $sp, -5
ADDI $v0, $zero, 4
ADDI $a0, $sp, 0
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

```
//Neil deGrasse Tyson
#include <iostream>
using namespace std;
int main()
{
    cout << "There is no "
         << "greater educ";
    cout << "ation\nthan one ";
    cout << "that is self-driven."
         << endl;
}
```

Output:

(b) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    int year=1, bal=1000, expenses=200;
    while( bal > 0 ) {
        cout << "Year " << year
             << ":  Balance: $"
             << bal << endl;
        bal = bal - expenses;
        year++;
    }
    return 0;
}
```

Output:

(c) What is the output:

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

      return 0;
}
```

Output:

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

C++ program:

```
#include <iostream>
using namespace std;
int main()
{
  int num =  1;
  while ((num < 0) || (num%2 == 1))
  {
    cout << "Enter small even #:";
    cin >> num;
  }
  cout << "Your number: " << num;
  return 0;
}
```

Python program:

(b) Write a C++ program that will ask for the time in 24 hour format (e.g. 2034 is 8:34pm) and, prints out "Morning Twilight" if the time is between 5am (e.g. 500) and 5:45am (e.g. 545), "Daylight" if the time is between 5:45am (e.g. 545) and 8pm (e.g. 2000) "Evening Twilight" if the time is between 8pm (e.g. 2000) and 8:30pm (e.g. 2030), and otherwise print "Night"

A sample run:

```
Enter time:  2015
Evening Twilight
```