Row:	Seat:

Final Exam, Version 3
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

20 December 2021

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:								

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]*

(Image from wikipedia commons)

1. (a) Given the quote in the code below, fill in the code to produce the Output on the right:

```
quote = ' "My life is my message." -- Mahatma Gandhi'
```

i.
```
print(quote[ ____ ])
```

Output:
```
Mahatma Gandhi
```

ii.
```
print(quote[2:4]. ________ )
```

Output:
```
my
```

iii.
```
print("This quote has", end=" ")
print(quote.count( ____ ), "period")
```

Output:
```
This quote has 1 period
```

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

i.
```
numbers = "10 / 11 / 12 / 13 / 14"
num_list = numbers. ________
```

ii.
```
for n in num_list :
    print( ________ )
```

Output:
```
0
1
0
1
0
```

(c) Consider the following shell commands:

```
$ ls
avg.py images logarithm.py logo.png reds.png
```

i. What is the output for:
```
$ ls | grep log
```

Output:

ii. What is the output for:
```
$ mkdir images/colors
$ mv logo.png images
$ mv reds.png images/colors
$ cd images
$ ls
```

Output:

iii. What is the output for:

```
$ cd ../
$ ls *.py
```

Output:

2. (a) Select the color corresponding to the rgb values below:

   i. `rgb = (255, 0, 255)`
   ☐ black ☐ red ☐ white ☐ gray ☐ purple

   ii. `rgb = "#000000"`
   ☐ black ☐ red ☐ white ☐ gray ☐ purple

   iii. `rgb = (0.5, 0.5, 0.5)`
   ☐ black ☐ red ☐ white ☐ gray ☐ purple

   iv. Select the SMALLEST Binary number:
   ☐ 110100 ☐ 011101 ☐ 101000 ☐ 000111 ☐ 101010

   v. What is the Hexadecimal number equivalent to decimal 15?
   ☐ 0F ☐ 99 ☐ A0 ☐ FF ☐ C3

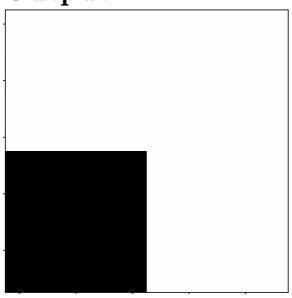
   (b) Given the list `names` below, fill in the code to produce the Output on the right:

```
names = [ "aida", "mandy", "diana", "roziena", "nancy", "lola" ]
```

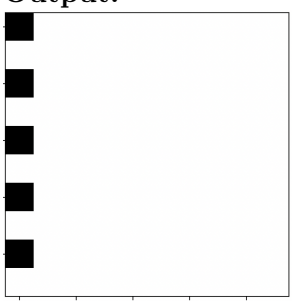
   i.
```
for i in range( [ ] ):
    print(names[i], end=" ")
```
Output:
```
aida mandy diana
```

   ii.
```
for j in range( [ ] , [ ] , [ ] ):
    print(names[j], end=" ")
```
Output:
```
mandy roziena lola
```

   iii.
```
import numpy as np
import matplotlib.pyplot as plt
im = np.ones( (10,10,3) )
im[ [ ] , [ ] , :]  = 0
plt.imshow(im)
plt.show()
```
Output:



   iv.
```
import numpy as np
import matplotlib.pyplot as plt
im = np.ones( (10,10,3) )
im[ [ ] , [ ] , :]  = 0
plt.imshow(im)
plt.show()
```
Output:

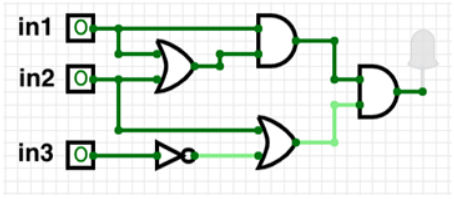


3. (a) What is the value (True/False):

i.
```
in1 = True
in2 = False
out = not in1 and in2
```
☐ True ☐ False

ii.
```
in1 = True
in2 = False
in3 = in1 or not in2
out = not(in1 or not in2) and not in3
```
☐ True ☐ False

iii.



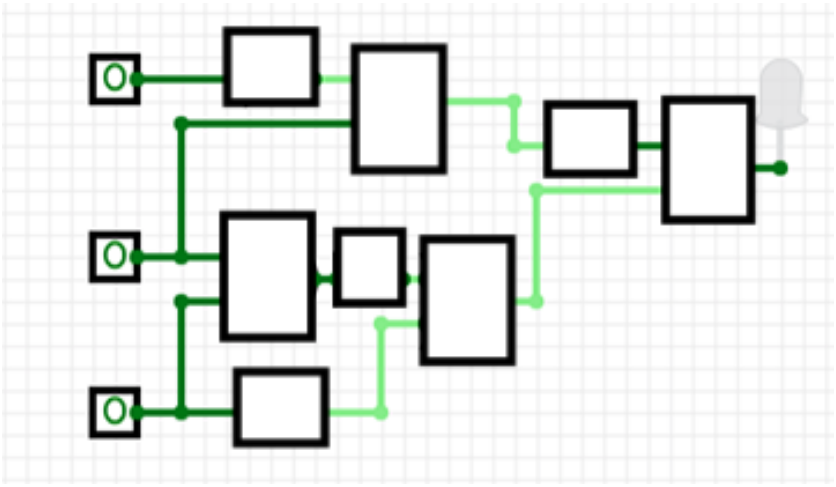

```
in1 = True
in2 = False
in3 = False
```
☐ True ☐ False

(b) Draw a circuit that implements the logical expression:

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

(c) Fill in the circuit with the gate-symbol or gate-name that implements the logical expression:

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



4. Consider the following functions:

```
def count(items, ch):
  count = 0
  for i in range(len(items)):
    if compare(items[i], ch):
        count += 1
  return count
```

```
def compare(i, c):
   return i == c


def main():
  chars = ['a','a','b','c','b','b','c','c']
  print(count(chars, 'a'))
```

(a) What are the formal parameters for `compare()`?

(b) What are the actual parameters for `count()`?

(c) How many calls are made to `compare()` after calling `main()`?

(d) What is the output after calling `main()`?

Output:

5. Design an algorithm that asks the user for the name of a text file containing a grid of numbers and loads it into a 2D array of integers (think like an image without the color channel), as well as an input number `n`. The program outputs the index `(row, col)` of the first occurrence of `n` in the array.

Libraries:

Input:

Output:

Design Pattern:
☐ Search ☐ Find Min ☐ Find Max ☐ Find All

Principal Mechanisms (select all that apply):
☐ Single Loop ☐ Nested Loop ☐ Conditional (if/else) statement
☐ Indexing / Slicing ☐ `split()` ☐ `input()`

Process (as a concise and precise LIST OF STEPS / pseudocode):
(Assume libraries have already been imported.)

6. Consider the `open_restaurants.csv` dataset for **restaurant reopening applications** under Phase Two of the New York Forward Plan to place outdoor seating in front of their business on the sidewalk and/or roadway. **Each row in the dataset corresponds to an application**. A snapshot of the data is given in the image below:

Seating Interest	Restaurant Name	Borough	Sidewalk Area	Roadway Area	Approved for Sidewalk Seating	Approved for Roadway Seating
sidewalk	HUNGRY GHOST	Manhattan	200	640	yes	no
both	Prince Laban&Chinese rest	Queens	144	144	yes	yes
sidewalk	Philly Pretzel Factory	Brooklyn	6500	920	yes	no
both	BICKLES TO GO	Bronx	100	160	yes	yes
roadway	STARBUCKS	Manhattan	160	160	no	yes
roadway	OVENLY	Brooklyn	40	168	no	yes
sidewalk	LE PAIN QUOTIDIEN	Manhattan	105	280	yes	no
both	Le Pain Quotidien GCW	Manhattan	90	240	yes	yes
both	Asian Kabab and Curry	Brooklyn	60	60	yes	yes

Fill in the Python program below:

```
#Import the libraries for data frames

[                                                    ]

#Prompt user for input file name:

csvFile = [                                          ]

#Read input data into data frame:

df = [                                               ]

#Print the 5 restaurant names with most applications
#Each row is an application, count the number of rows
# per Restaurant Name and print the top 5

print( [                                             ] )

#Group the data by Borough to extract applications in Brooklyn
#use groupby and get_group

brooklyn = [                                         ]

#Print the average sidewalk area in Brooklyn

print( [                                             ] )
```

7. Consider the Python program below to display the first `n` squares. **Fill-in the functions** based on the comments and the overall program. Pay attention to the sample output in the comments in-order to implement the function correctly.

```
# Display's numbers 1 - n squared
# Example output for n = 5:
# 1 **2 = 1
# 2 **2 = 4
# 3 **2 = 9
# 4 **2 = 16
# 5 **2 = 25

def print_n_squares(n):
```

```


```

```
# Validate the input to be positive
# If the input is not positive,
# keep asking for the number.
# Example output:
# Please enter a positive number.
# How many squared numbers to display?

def validate_input(num):
```

```


```

```
# Display numbers squared
def main():
  i = int(input("How many squared numbers to display? "))
  i = validate(i)

  #print first n squares
  print_n_squares(i)
```

8. (a) What does the MIPS program below print:

Output:

(b) Modify the program to print out `Hell!`
Shade in the box for each line or line-pair that needs to be changed and rewrite the instruction below. If the line needs to be deleted, write `Delete`.

☐ `ADDI $sp, $sp, -7`

☐
```
ADDI $t0, $zero, 72       # store 72 in $t0
SB $t0, 0($sp)
```

☐
```
ADDI $t0, $zero, 101      # store 101 in $t0
SB $t0, 1($sp)
```

☐
```
ADDI $t0, $zero, 108      # store 108 in $t0
SB $t0, 2($sp)
```

☐
```
ADDI $t0, $zero, 108      # store 108 in $t0
SB $t0, 3($sp)
```

☐
```
ADDI $t0, $zero, 111      # store 111 in $t0
SB $t0, 4($sp)
```

☐
```
ADDI $t0, $zero, 33       # store 33 in $t0
SB $t0, 5($sp)
```

☐
```
ADDI $t0, $zero, 0        # (null)
SB $t0, 6($sp)
```

☐ `ADDI $v0, $zero, 4        # 4 is for print string`

☐ `ADDI $a0, $sp, 0          # Set $a0 to stack pointer`

☐ `syscall                   # Print to the log`

(c) Modify the MIPS program below to count from 20 to 5, down by 5. Shade in the box for each line that needs to be changed and rewrite the instruction below.

☐ `ADDI $s0, $zero, 30 #set s0 to 30`

☐ `ADDI $s1, $zero, 3 #set s1 to 3`

☐ `ADDI $s2, $zero, 15 #use to compare for branching`

☐ `AGAIN: SUB $s0, $s0, $s1`

☐ `BEQ $s0, $s2, DONE`

☐ `J AGAIN`

☐ `DONE: #To break out of the loop`

(d) After the modification, how many times is the line labeled `AGAIN:` executed?

9. Fill in the C++ programs below to produce the Output on the right.

(a)

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

    for(int i = 3; i <=15;          ){
       cout << i+2  << endl;
    }
    return 0;
}
```

Output:

```
5
8
11
14
17
```

(b)

```
#include <iostream>
using namespace std;
int main()
{
    int count = 20;
    int num = 10;

    while(count >=0 && num          ){
        cout << count << " " << num << endl;
        count -= 2;
        num -= 5;
    }
    return 0;
}
```

Output:

```
20 10
18 5
```

(c)

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

    for (int i = 10;          ; i--){
         cout << "Keep going!" << endl;
    }
    return 0;
}
```

Output:

```
Keep going!
Keep going!
Keep going!
Keep going!
Keep going!
```

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

```
for i in range(2,13,2):
  for j in range(10,i,-3):
    print(i, j)
```

```
//include library and namespace
```

```
//main function signature
```

```
{
  //outer loop line
```

```
  //inner loop line
```

```
  //loop body
```

```
  //return
```

```
}
```

(b) Write a **complete C++ program** that asks the user for an amount and outputs the interest as follows:

- `"5%"` if the amount is less than $500
- `"8%"` if the amount is in range [$500 , $2000]
- `"10%"` otherwise

```
//include library and namespace
```

```
//main function signature
```

```
{
  //declare variables
```

```
  //obtain input
```

```
  //output interest
```

```
  //return
```

```
}
```