Row:	Seat:

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

13 December 2022

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

```
seasons = "Spring,Summer,Autumn,Winter"
```

i.
```
spring_summer = [          ]
for s in spring_summer:
    print( [          ] )
```

Output:
```
spring
summer
```

ii.
```
summer_winter = [          ]
for s in summer_winter:
    print( [          ] )
```

Output:
```
SUMMER
WINTER
```

(b) Consider the following shell commands:

```
$ pwd
/usr/student
$ ls
covid.csv grades.csv  happy.py  hello.py
```

i. What is the output for:
```
$ mkdir projects
$ mv *py projects
$ cd projects
$ ls
```

Output:

ii. What is the output for:
```
$ pwd
```

Output:

iii. What is the output for:
```
$ cd ..
$ ls | grep csv
```

Output:

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

   i. `rgb = (55, 55, 55)`
   ☐ black ☐ red ☐ white ☐ gray ☐ purple

   ii. `rgb = "#AB0000"`
   ☐ black ☐ red ☐ white ☐ gray ☐ purple

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

   iv. What is the binary number equivalent of decimal number 45?

   `Decimal 45 = Binary` [ ][ ][ ][ ][ ][ ]

   v. What is the Decimal number equivalent to Hexadecimal AC?

   `Hexadecmal AC = Decimal` [ ][ ][ ]

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

```
fruits = ['apple', 'bananna', 'coconut', 'dragon fruit', 'elderberry']
```

i.
```
for i in range( [ ] ):
    for j in range( [ ] ):
        print(fruits[j][ ], end=" ")
```

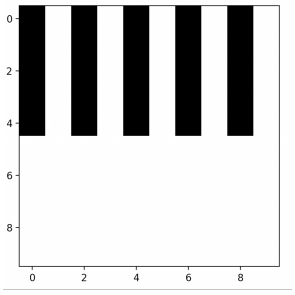
Output:
```
a b c d e
a b c d e
```

ii.
```
for j in range( [ ] , [ ] , [ ] ):
    print(fruits[j][ ], end=" ")
```

Output:
```
y t e
```

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

Output:



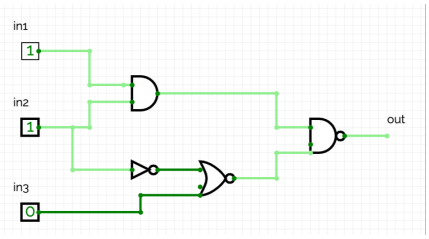
3. (a) What is the value (True/False):

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

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

iii.
```
in1 = True
in2 = False
in3 = not in1 or in2
out = not in1 and not in3
```
☐ True ☐ False

iv.




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

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

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

4. Consider the following functions:

```
def count(mylist, target):
    num_occur = 0
    for num in mylist:
        if equal(num, target):
            num_occur += 1

    return num_occur
```

```
def equal(a, b):
    return a == b

def main():
    mylist = [1, 6, 5, 7, 7]
    print(count(mylist, 6))
```

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

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

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

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

i. **Output:**

5. Design an algorithm that, given an image, outputs an image that make each pixel its complement. For a pixel with color (r, g, b), its complement color is (1-r, 1-g, 1-b). For example, if a pixel is 100% red, that is, (1, 0, 0), then its complementary color is (0, 1, 1).

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()` ☐ `groupby()`

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

6. Consider the `courses_training.csv` dataset that reports training courses offered in NY state. A snapshot given in the image below:

Organization	Borough	course name	Cost Total	Duration
1st Choice Ca	Brooklyn	Home Health	550	83
A.L.M. Securi	Brooklyn	8 HOUR PRE-	500	27
A.L.M. Securi	Brooklyn	16 HOUR OJT	822	20
ACCESS INST	Queens	ESL (Full Pro	4000	750
ACCESS INST	Queens	Home Health	750	83
ACCESS INST	Queens	Medical Assi	6000	600

Fill in the Python program below:

```
#Import the libraries for data frames.
```

```
#Read input data into data frame:
```

```
df =
```

```
#Calculate hourly_rate by dividing Cost Total by Duration (in hours)
```

```
#Groups the data by Borough to extract data in Queens.
```

```
queens =
```

```
#Print the minimum, maximum, and average hourly_rate of all training courses in Queens.
```

7. Write a **complete Python program** that prompts the user for the name of a .csv file. Suppose column name of longitude is Longitude and column name for latitude is Latitude and generates an interactive .html map with markers found at each geographical location extrated from the .csv file.

```
#Import the packages for dataframes and for generating html maps
```

```
#Ask user for the name of csv file and store in variable in_file
```

```
#Read the csv file into a dataframe and store it in variable df
```

```
#Create a map and store in variable map
```

```
#Loop through all the rows in the dataframe, create a marker with
#values found in columns lat and long, add marker to the map
```

```
#Save the map to file named map.html
```

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

Output:

(b) Modify the program to print out 6 consecutive letters in decreasing order ('Z' down to 'U'). Shade in the box for each line that needs to be changed and rewrite the instruction below.

☐ `ADDI $sp, $sp, -4 # Set up stack`

☐ `ADDI $t0, $zero, 97 # Set $t0 at 97 (a)`

☐ `ADDI $s2, $zero, 3 # Use to test when you reach 3`

☐ `SETUP: SB $t0, 0($sp) # Next letter in $t0`

☐ `ADDI $sp, $sp, 1 # Increment the stack`

☐ `ADDI $s2, $s2, -1 # Decrement the counter by 1`

☐ `ADDI $t0, $t0, 2 # Increment the letter by two`

☐ `BEQ $s2, $zero, DONE # Jump to DONE if s2 == 0`

☐ `J SETUP # Else, jump back to SETUP`

☐ `DONE: ADDI $t0, $zero, 0 # Null (0) to terminate string`

☐ `SB $t0, 0($sp) # Add null to stack`

☐ `ADDI $sp, $sp, -3 # Set up stack to print`

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

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

☐ `syscall # Print to the log`

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 = 7; i <= [          ] ;
[          ]){
       cout << i+2  << endl;
    }
    return 0;
}
```

Output:

```
9
12
15
```

(b)

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

    while(count >=0 && num [      ]){
        cout << count << " " << num << endl;
        count -= 5;
        num -= 4;
    }
    return 0;
}
```

Output:

```
20 10
15 6
10 2
```

(c)

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

    for (int i = 9; [          ] ; i--){
         cout << "Keep going!" << endl;
    }
    return 0;
}
```

Output:

```
Keep going!
Keep going!
Keep going!
Keep going!
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(1, 10):
    for j in range(1, i+1):
        print(i*j, end=' ')
    print()
```

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

```
//main function signature
```

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

```
  {
  //inner loop line
```

```
  //loop body
```

```
  }
  //return
```

```
}
```

(b) One gallon is 3.78541 liters, it is also equal to 128 oz.
Write a **complete C++ program** that asks the user for the number of gallons and prints the corresponding number of liters and oz.

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

```
//main function signature
```

```
{
  //initialize variables
```

```
  //obtain input
```

```
  //calculate conversions
```

```
  //output conversions
```

```
  //return
```

```
}
```