Row:	Seat:

Final Exam, Version 4
CSci 127: Introduction to Computer Science
Hunter College, City University of New York

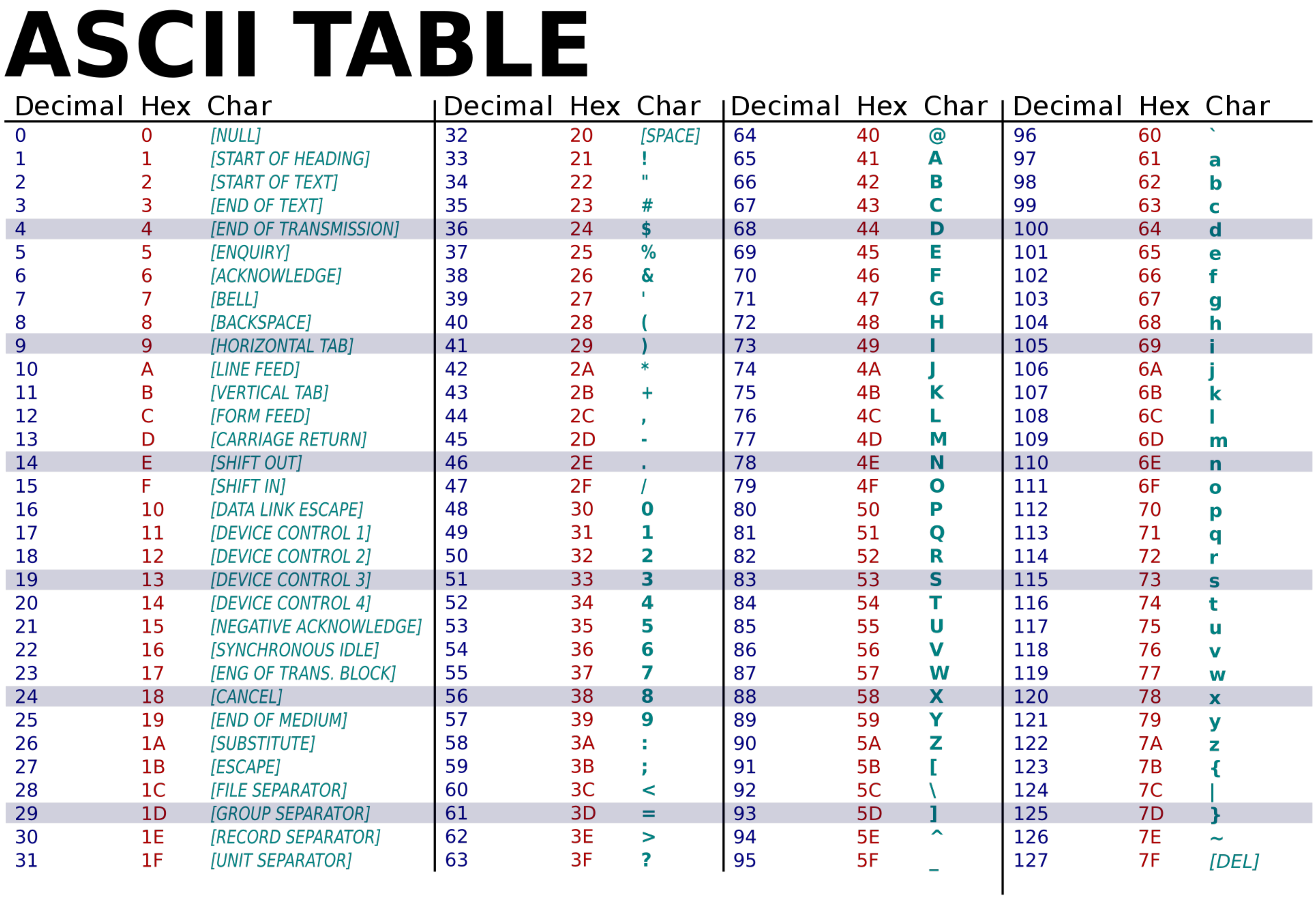
20 December 2021

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 = ' "Simplicity is the ultimate sophistication." Leonardo da Vinci'
```

i.

```
print( □ )
```

Output:

```
Leonardo
```

ii.

```
print(quote □ )
```

Output:

```
simplicity
```

iii.

```
words = □
print("This quote has", len(words)-4, "words")
```

Output:

```
This quote has 5 words
```

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

```
letters = "z * y * x * w"
```

i.

```
print("There are", letters. □ , "letters")
```

ii.

```
for i in range(len(letters)):
    if □ :
        print(letters[i])
```

Output:

```
There are 4 letters
z
y
x
w
```

(c) Consider the following shell commands:

```
$ ls
code web
```

i. What is the output for:

```
$ cd code
$ ls
plots   star.py    turtle_progs
$ mv star.py turtle_progs/
$ ls
```

Output:

□

ii. What is the output for:

```
$ cd turtle_progs/
$ ls
panorama.py  ramble.py  star.py
$ ls | grep ra*
```

Output:

□

iii. What is the output for:

```
$ cd ../ ../
$ ls
```

Output:

□

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

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

ii. `rgb = "#ABABAB"`
□ black □ red □ white □ gray □ purple

iii. `rgb = (0.0, 0.0, 0.0)`
□ black □ red □ white □ gray □ purple

iv. Select the LARGEST Binary number:
□ 110100 □ 011101 □ 101000 □ 000111 □ 101010

v. What is the Binary number equivalent to decimal 160?
□ 0F □ 99 □ A0 □ FF □ C3

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

```
symbols = [ "*", "#", "+", "$"]
```

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

Output:

```
* # + $
```

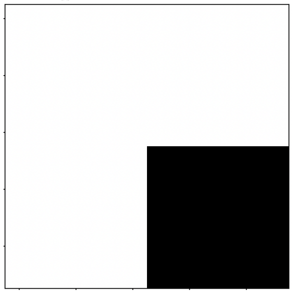
ii.
```
for j in range( □ , □ , □ ):
   print(symbols[j], end=" ")
```

Output:

```
* $
```

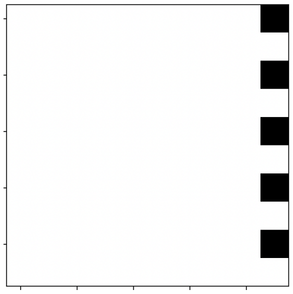
iii.
```
import numpy as np
import matplotlib.pyplot as plt
im = np.ones( (10,10,3) )
im[□ , □ , :]  = 0
plt.imshow(im)
plt.show()
```

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 or  in2)
```

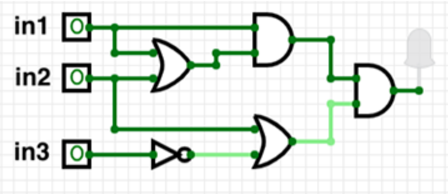
□ True □ False

ii.

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

□ True □ False

iii.



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

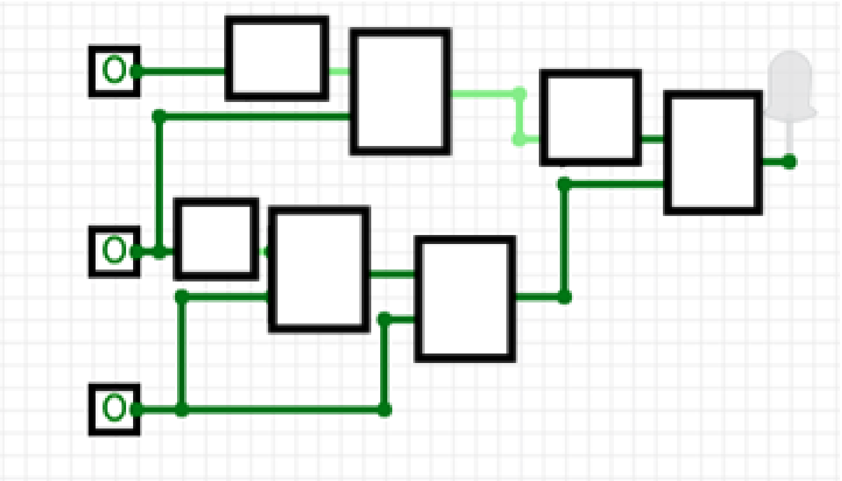
□ True □ False

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

```
(in1 or in2) and 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 in3)
```



4. Consider the following functions:

```
def add_odd(items):
  sum = 0
  for i in range(len(items)):
    sum += compare(items[i])
  return sum
```

```
def compare(i):
  return i * (i % 2)


def main():
  nums = [1, 2, 3, 4, 5, 6, 7, 8, 9]
  print(add_odd(nums))
```

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

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

(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 number of occurrences of `n` found in the grid.

Libraries:

Input:

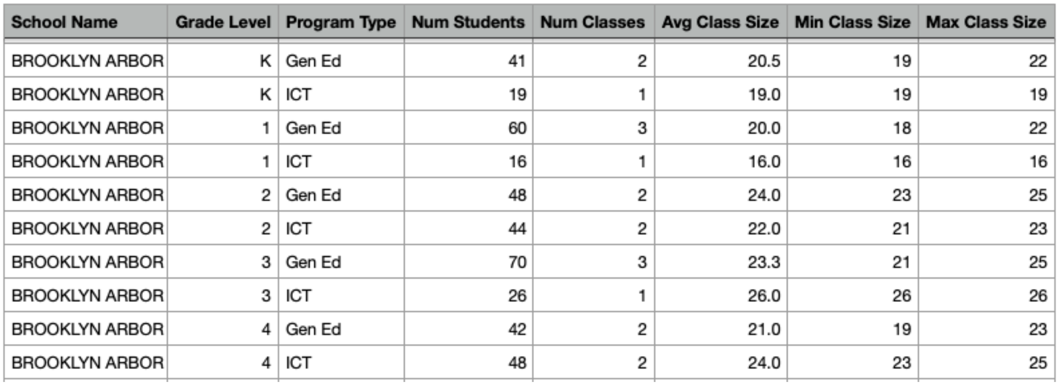
Output:

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 `class_size.csv` dataset from NYC Open Data **preliminary average class size aggregated by school for 2021**. Each row in the dataset corresponds to a class grade level and program type at a given school. A snapshot of the data is given in the image below:



School Name	Grade Level	Program Type	Num Students	Num Classes	Avg Class Size	Min Class Size	Max Class Size
BROOKLYN ARBOR	K	Gen Ed	41	2	20.5	19	22
BROOKLYN ARBOR	K	ICT	19	1	19.0	19	19
BROOKLYN ARBOR	1	Gen Ed	60	3	20.0	18	22
BROOKLYN ARBOR	1	ICT	16	1	16.0	16	16
BROOKLYN ARBOR	2	Gen Ed	48	2	24.0	23	25
BROOKLYN ARBOR	2	ICT	44	2	22.0	21	23
BROOKLYN ARBOR	3	Gen Ed	70	3	23.3	21	25
BROOKLYN ARBOR	3	ICT	26	1	26.0	26	26
BROOKLYN ARBOR	4	Gen Ed	42	2	21.0	19	23
BROOKLYN ARBOR	4	ICT	48	2	24.0	23	25

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 number of rows per Program Type
# (i.e. number of rows for Gen Ed, number of rows for ICT, etc.)

print(                                                              )

#Group the data by Grade Level to extract Kindergarten
#use groupby and get_group

kindergarten =

#Print the average class size for kindergarten across all schools

print(                                                              )
```

7. Consider the Python program below to display the first n Fibonacci numbers. The Fibonacci sequence is generated as follows: F0 = 0, F1 = 1, F2 = F1 + F0, F3 = F2 + F1, ... , Fn = Fn-1 + Fn-2. **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.

```
# Displays n Fibonacci numbers
# Example output for n = 7:
# F0 = 0
# F1 = 1
# F2 = 1
# F3 = 2
# F4 = 3
# F5 = 5
# F6 = 8
# F7 = 13
def print_n_fib(n):
```

```
# Validate the input to be > 2
# If the input is not > 2,
# keep asking for the number.
# Example output:
# Please enter a number > 2.
# How many Fibonacci numbers to print?

def validate_input(num):
```

```
# Display n Fibonacci numbers
def main():
  n = int(input("How many Fibonacci numbers to print? "))
  n = validate(n)

  #print n Fibonacci numbers
  print_n_fib(n)
```

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

   Output:

   (b) Modify the program to print out `HELLO`
   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 10 to 30, up 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 = 0; □ i += 10){
       cout << i*2  << endl;
    }
    return 0;
}
```

Output:

```
0
20
40
60
```

(b)

```
#include <iostream>
using namespace std;
int main()
{
    int count = 0;
    int num = 0;

    while(count □ && num □ ){
        cout << count << " " << num << endl;
        count +=1;
        if(count % 2 == 0)
            num +=1;
    }
    return 0;
}
```

Output:

```
0 0
1 0
2 1
3 1
4 2
5 2
```

(c)

```
#include <iostream>
using namespace std;
int main(){
    for (int i = 5; □ i--){
         cout << "Keep going!" << endl;
    }
    return 0;
}
```

Output:

```
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(20,3,-5):
  for j in range(50,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 the number of credit hours and outputs the student category on a new line as follows:

- `"Freshman"` for [0,29] hours of earned credit
- `"Sophomore"` for [30,59] hours of earned credit
- `"Junior"` for [60,89] hours of earned credit
- `"Senior"` 90 or more hours of earned credit

```
//include library and namespace

//main function signature

{
  //declare variables

  //obtain input

  //output student category

  //return

}
```