

Row:	SEAT:

FINAL EXAMINATION, VERSION 2
 CSci 127: Introduction to Computer Science
 Hunter College, City University of New York
 May 2026

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) Assuming the code is run sequentially, what is the output at each step:

```

s = "elion,gertrude;cohn,mildred;\
    petters,arlie"
i. a = s[-5:]
   print(a.upper())

```

Output:

```

ii. names = s.split(';')
     print(names[-1])

```

Output:

```

iii. firsts = [n[0] for n in names]
      print(firsts)

```

Output:

```

iv. for n in names:
     w = n.split(',')
     print(w[1],w[0])

```

Output:

- (b) Consider the following shell commands:

```

$ ls
cheese          pepperoni      peppers        pesto_sauce    tomato_sauce
$ pwd
/Users/pizza

```

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

```

i. $ mkdir sauce
   $ mv *_* sauce
   $ ls

```

Output:

```

ii. $ cd sauce
     $ echo "Sauce options:"
     $ ls

```

Output:

```

iii. $ cd ../
      $ pwd

```

Output:

```

iv. $ echo "Toppings:"
     $ ls *e* | wc -l

```

Output:

2. (a) Check **all that apply**:

i. Fill in the code below to change the turtle, `ryan`, to be the brightest green:

```
ryan.color("#       ")
```

ii. Select all the **even binary numbers**:

1000 1101 0111 1010 1111

iii. Select all the **hexadecimal numbers larger than 16 decimal**:

A B 90 99 FF

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

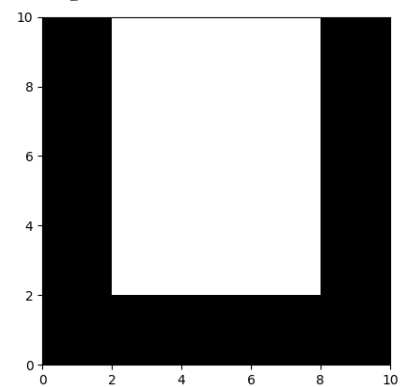
```
import matplotlib.pyplot as plt
import numpy as np
#Set background color to white:

logo =  ( (10,10,3) )
#Draw letter in black:
logo[:,2,:] = 0

logo[:, ,:] = 0

logo[ ,:,:] = 0
plt.imshow(logo)
plt.show()
```

Output:



(c) For each error below, give the line number and the code that would fix the error.

```
1 import turt
2 tia = Turtle.Turtle()
3 for i in range(5:
4     tia.right(360/5)
5     tia.forward(20)
```

i. `import turt`
 `ModuleNotFoundError: No module named 'turt'`

Line Number: **Code that fixes the error:**

ii. `tia = Turtle.Turtle()`
 `~~~~~`
 `NameError: name 'Turtle' is not defined.`

Line Number: **Correct code:**

iii. `for i in range(5:`
 `^`
 `SyntaxError: invalid syntax`

Line Number: **Correct code:**

3. (a) What will make the following statement true:

`not (in1 or in2)`

- | | |
|--|---|
| <input type="checkbox"/> Setting <code>in1 = False</code> and <code>in2 = False</code> . | <input type="checkbox"/> Setting <code>in1 = True</code> and <code>in2 = False</code> . |
| <input type="checkbox"/> Setting <code>in1 = False</code> and <code>in2 = True</code> . | <input type="checkbox"/> Setting <code>in1 = True</code> and <code>in2 = True</code> . |
| <input type="checkbox"/> All values for <code>in1</code> and <code>in2</code> make the statement true. | <input type="checkbox"/> No values for <code>in1</code> and <code>in2</code> make the statement true. |

(b) What is the value of `out`?

`in1 = True`
`in2 = False`
`out = not in1 or not in2`

Choose one:

- False Not defined
 True

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

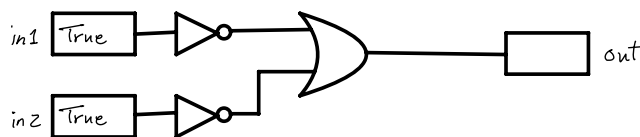
`in1 =`

`in2 =`

`out =`

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

(d) What is the output of this circuit?

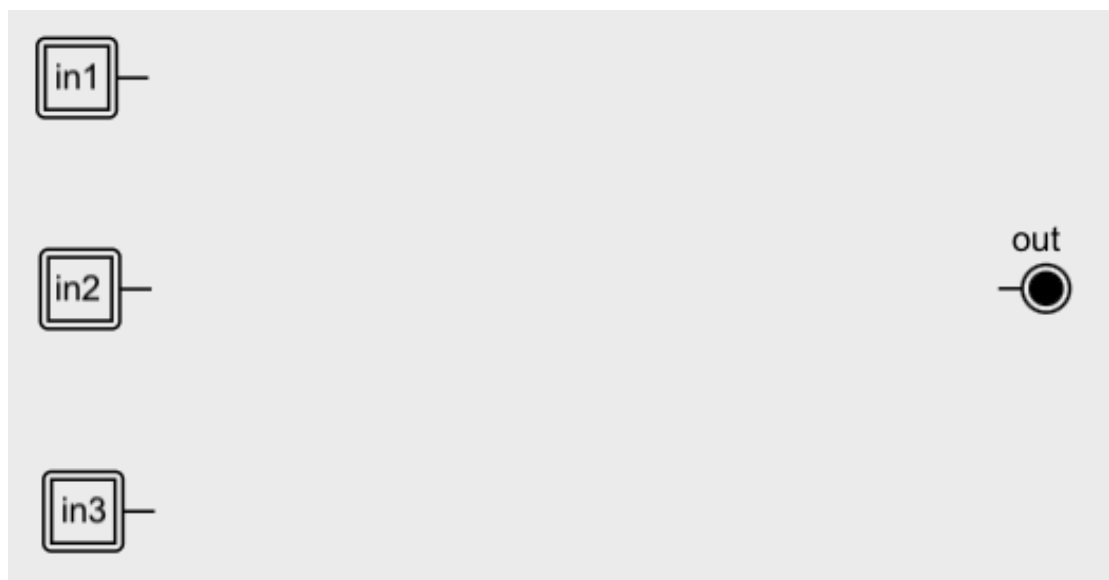


Choose one:

- False Not defined
 True

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

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



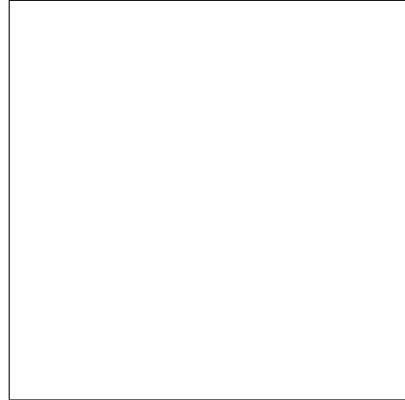
4. (a) Draw the output for the function calls:

```

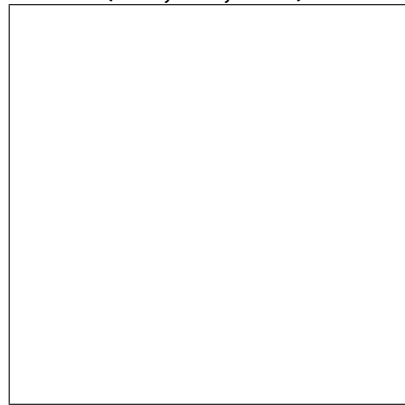
1 import turtle
2 tom = turtle.Turtle()
3 tom.shape('triangle')
4
5 def ramble(ty, dist, stamp):
6     if dist > 10:
7         for i in range(3):
8             ty.left(120)
9             ty.forward(dist)
10            ramble(ty,dist//2,stamp)
11        elif stamp:
12            for i in range(3):
13                ty.forward(20)
14                ty.stamp()
15        else:
16            ty.forward(20)

```

i. `ramble(tom,8,False)`



ii. `ramble(tom,100,True)`



(b) What are the formal parameters for `ramble()`:

(c) If you call `ramble(tom,8,False)`, which branches of the function are tested (check all that apply):

- The block of code at Lines 7-10.
- The block of code at Lines 12-14.
- The block of code at Line 16.
- None of these blocks of code (lines 7-10, 12-14, 16) are visited from this invocation (call).

(d) If you call `ramble(tom,100,True)`, which branches of the function are tested (check all that apply):

- The block of code at Lines 7-10.
- The block of code at Lines 12-14.
- The block of code at Line 16.
- None of these blocks of code (lines 7-10, 12-14, 16) are visited from this invocation (call).

5. Design an algorithm that takes a string and returns the most common word in the string. If there is not a unique word that occurs most often, return the first one alphabetically. Your algorithm, if given the input:

"A rose is a rose is a rose. -Gertrude Stein"

would return since **a** since a, rose, and is all occur 3 times but a is first alphabetically.

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 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,
- create a new column that is double the column that the user provided,
- print out the mean value of the double column, and
- displays a plot of the double column (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_name =
```

```
#Read in the CSV file to a DataFrame:
```

```
df =
```

```
#Create a new column that's double that of the column specified:
```

```
df["Double"] =
```

```
#Compute the average value of the column, "Double":
```

```
print("Mean of column 'Double' is", m)
```

```
#Display a plot of "Year" vs. "Double" column
```

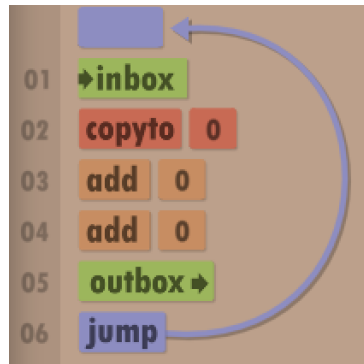
7. Write a **complete program** that identifies all two-part numbers from a string and returns a list of those numbers. In the input string, each number is separated by a comma followed by a space: ', ', and two-part numbers contain a '-'.

For example, if you ran your program and the user entered:

Enter numbers, separated by commas: twenty, four, thirty-one, sixty-six, eight

A list of the two-part numbers printed is: ["thirty-one", "sixty-six"].

8. (a) What does the Human Resource Machine (HRM) code output with the following input:



Output:

- (b) Consider the following MIPS program:

```

ADDI $s0, $zero, 10
ADDI $s1, $s0, 5
ADD $s2, $s1, $s1
SUB $s3, $s0, $s2

```

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

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

- (c) Consider the MIPS code:

```

1  ADDI $sp, $sp, -9
2  ADDI $t0, $zero, 33
3  ADDI $s2, $zero, 8
4  SETUP: SB $t0, 0($sp)
5  ADDI $sp, $sp, 1
6  ADDI $s2, $s2, -1
7  BEQ $s2, $zero, DONE
8  J SETUP
9  DONE: ADDI $t0, $zero, 0
10 SB $t0, 0($sp)
11 ADDI $sp, $sp, -8
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 message printed?	
iv) List what you need to change to print half of the message:	

9. (a) Fill in the missing code to yield the output:

```
//Quote by George R.R. Martin
#include <iostream>
using namespace std;
int main()
{
    cout << "A mind needs books ";
    
    cout << "a whetstone," << endl;
    cout << "if it is to keep its edge.";
    return 0;
}
```

Output:

A mind needs books as
a sword needs a whetstone,
if it is to keep its edge.

- (b) What is the output:

```
//More Game of Thrones
#include <iostream>
using namespace std;
int main()
{
    int count = 3;
    while (count > 1) {
        cout <<"Winter is coming ";
        count--;
    }
    cout << "!\nNothing burns ";
    cout << "like the cold." << endl;
    return 0;
}
```

Output:

- (c) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 0; i < 5; i++)
    {
        cout << endl << i;
        for (int j = 1; j < 5; j++)
        {
            if (i % 2 == 0)
                cout << "a";
            else
                cout << "b";
        }
    }
    return 0;
}
```

Output:

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

C++ program:

```
#include <iostream>
using namespace std;
int main()
{
    float fah;
    cout<<"Enter temperature in fahrenheit: ";
    cin >> fah;
    float cel = ((5.0/9.0)* (fah - 32));
    cout << cel << endl;
    return 0;
}
```

Python program:



- (b) Write a **complete C++ program** that asks for a positive whole number, **num**, and prints out the partial sum up to **num** (e.g. $1, 1 + 2, 1 + 2 + 3, \dots, 1 + 2 + \dots + num$).

A sample run of your code:

Enter num: 5

1

3

6

10

15

SCRATCH PAPER

SCRATCH PAPER

CSCI 127 Reference Sheet, Spring 2026

Turtles: Let *t* be a turtle.

Function	Description
<code>t=turtle.Turtle()</code>	Creates turtle <i>t</i> .
<code>t.forward(x)</code>	Moves <i>t</i> forward <i>x</i> steps.
<code>t.backward(x)</code>	Moves <i>t</i> back <i>x</i> steps.
<code>t.left(x)/t.right(x)</code>	Turns <i>t</i> left/right <i>x</i> deg.
<code>t.penup()/t.pendown()</code>	Lifts <i>t</i> 's pen up/down.
<code>t.stamp()</code>	Stamps at <i>t</i> 's location.
<code>t.goto(x,y)</code>	Moves <i>t</i> to (<i>x</i> , <i>y</i>).
<code>t.colormode(255)</code>	Enable 255 mode.
<code>t.color(s)/t.color(x,y,z)</code>	Changes <i>t</i> 's color.

String Methods: Let *s* be a string.

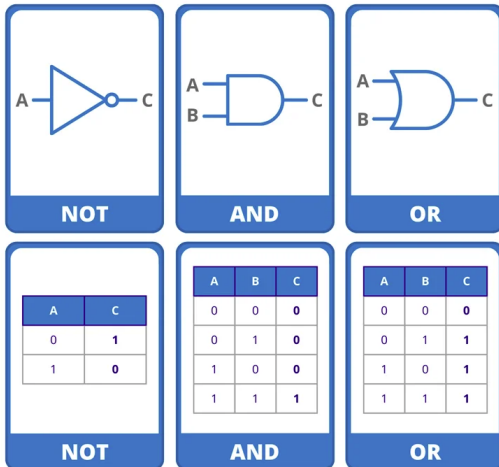
Function	Description
<code>len(s)</code>	Returns the length of <i>s</i> .
<code>s.lower()</code>	Returns <i>s</i> as lower case character(s).
<code>s.upper()</code>	Returns <i>s</i> as upper case character(s).
<code>s.count(t)</code>	Returns count of <i>t</i> in <i>s</i> .
<code>s.find(t)</code>	Returns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

Containers: Lists, Ranges & Dictionaries.

Function	Description
<code>l = []</code>	Creates an empty list.
<code>l = [a,b,c]</code>	List with 3 elements.
<code>l.append(elt)</code>	Adds <i>elt</i> to end of list.
<code>l[i]</code>	Access element at index <i>i</i> .
<code>range(start,stop,step)</code>	Range object from <i>start</i> to <i>stop</i> -1, by <i>step</i> .
<code>zip(l1,l2)</code>	Combines <i>l1</i> & <i>l2</i> pairwise.
<code>[x*x for x in l1]</code>	List of <i>l1</i> 's elements squared. (using list comprehension).
<code>d = {}</code>	Creates an empty dictionary.
<code>d = {k1:v1,k2:v2}</code>	Dictionary of key/value pairs.
<code>d[k] = v</code>	Adds <i>k</i> : <i>v</i> to dictionary.
<code>d[k]</code>	Access value at key <i>k</i> .
<code>k in d</code>	Checks if key is in dictionary.
<code>d.keys() / d.values()</code>	Returns keys/values of <i>d</i> .

Functions:

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



(from truthtablegen.com)

Control Structures:

Function	Description
<code>for i in range(0,100,5):</code>	Basic for-loop.
<code>for c in str_var:</code>	Looping through a string.
<code>if x==0:... elif x>0:... else:... </code>	Basic if-statement.

numpy: Let *np* be the numpy package.

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

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

Function	Description
<code>pd.read_csv(fn)</code>	Returns a DataFrame with file <i>fn</i> .
<code>pd.DataFrame(d)</code>	Returns DataFrame of dictionary <i>d</i> .
<code>df.to_csv(file_name)</code>	Writes <i>df</i> to <i>file_name</i> .
<code>df[col]</code>	Returns <i>col</i> column as a Series.
<code>df[[c1,c2]]</code>	Returns DataFrame with <i>c1</i> & <i>c2</i> .
<code>df.columns</code>	List of column names of <i>df</i> .
<code>df.head(n)/df.tail(n)</code>	First/last <i>n</i> lines of <i>df</i> .
<code>df.plot(x=col)</code>	Returns a figure with <i>col</i> as x-axis.
<code>fig.savefig(fn)</code>	Writes <i>fig</i> to <i>fn</i> .
<code>s.min()/s.max()/s.mean()</code>	Returns min/max/average of <i>s</i> .
<code>s.value_counts()</code>	Counts # times each value occurs.
<code>df.groupby(col)</code>	Groups <i>df</i> by values in <i>col</i> .
<code>df.dropna(subset=[c1,c2])</code>	Drops rows with missing <i>c1</i> or <i>c2</i> values.

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

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

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

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

UNIX:

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

C++:

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

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]