Row:	Seat:

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

Fall 2025

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

```
stops = "Grand Central-42 St,51 St,59 St,68 St-Hunter College"
```

	Python Code:	**Output:**
i.	`print(stops[ ])`	`Gran`

ii.	`counts = {}`
	`for c in stops.upper():`
	`  if :`
	`  counts[c] = counts[c]+1`
	`  else:`
	`  counts[c]=1`
	`print("L appears", counts['L'], "times.")`
		`L appears 3 times.`

iii.	`s_list = stops.` `print(s_list[-1].upper())`	`68 ST-HUNTER COLLEGE`
iv.	`len_s = [ for st in s_list]` `print(len_s[1:3])`	`[5, 5]`
v.	`min_l = min( )` `print("Length of shortest name is", min_l)`	`Length of shortest name is 5`

(b) The commands below are **run sequentially**, what is the output after each has run:

```
$ ls -l
-rw-r--r--  1 stjohn  staff    339 Nov  3 16:02 p4.py
-rw-r--r--  1 stjohn  staff    379 Nov 13 15:31 p5.py
-rw-r--r--  1 stjohn  staff    480 Dec  7 14:43 p6V0.py
-rw-r--r--  1 stjohn  staff    736 Dec  3 13:09 p7v0.py
-rw-r--r--  1 stjohn  staff    141 Dec  6 21:01 p9a.cpp
-rw-r--r--  1 stjohn  staff    237 Dec  7 15:14 p9b.cpp
$ pwd
/tmp/v4
```

i.
```
$ mkdir python_progs
$ mv *.py python_progs
$ ls *.cpp
```

Output:

ii.
```
$ echo "Program 9:"
$ ls | grep p9
```

Output:

iii.
```
$ cd python_progs
$ pwd
$ ls -l | grep Nov | wc -l
```

Output:

2. (a) Fill in the missing values in the table:

Decimal	Binary	Hexadecimal
2		2
	1100	C
32	100000	
253	11111101	

(b) Fill in the missing information to make the statements true:

```
import turtle
john = turtle.Turtle()
turtle.colormode(1.0)
john.color(0, [          ])
faye = turtle.Turtle()
faye.color("#AAAAAA")
sara = turtle.Turtle()
turtle.colormode(255)
sara.color(200, 0, 200)
zee = turtle.Turtle()
zee.color("# [          ]")
ping = turtle.Turtle()
turtle.colormode(255)
ping.color(200, 0, 0)
```

i. [zee] is red.

ii. [ ] is purple.

iii. [john] is blue.

iv. [ ] is gray.

v. [ ] is bright pink.

(c) Consider the code:

```
1 words = "
2 while words == "" or len(words) > 10
3     words = input('Enter a string with 1 to 10 characters: ')
4 print('You entered:'  words)
```

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

```
    words = "
            ^
SyntaxError: unterminated string literal (detected at line 1)
```

Write the code that would fix the error:

[ ]

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

```
    while words == "" or len(words) > 10
                                     ^
SyntaxError: expected ':'
```

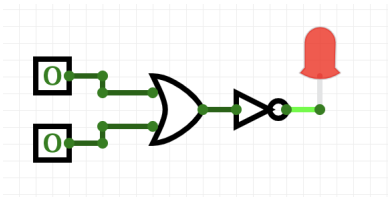
Write the code that would fix the error:

[ ]

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

i.
```
in1 = True
in2 = False
out = in1 or in2
```
out = ____

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

iii. 

out = ____

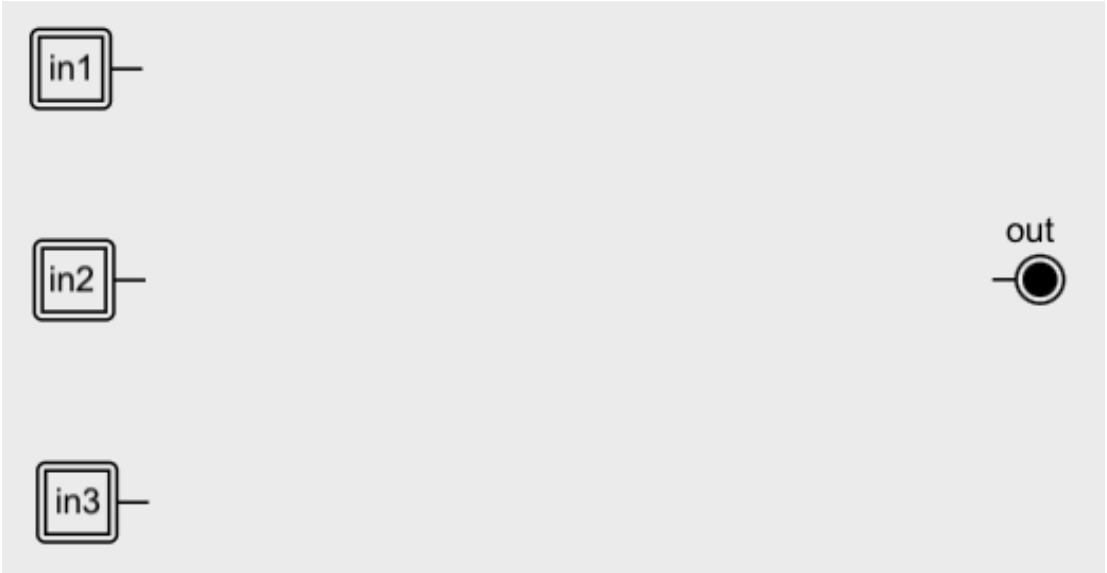
```
in1 = True
in2 = False
```

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

i.
```
in1 = ____
in2 = ____

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

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

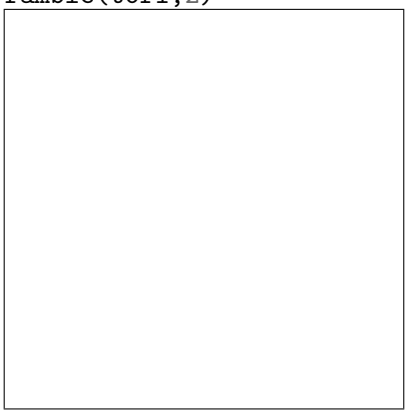
```
not ((in1 and in2) or (in1 and (in2 or in3)))
```



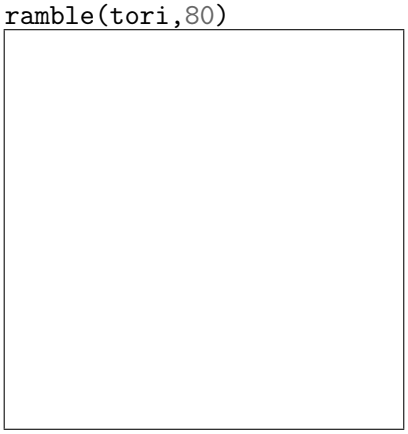
4. (a) Draw the output for the function calls:

```
1  import turtle
2  tori = turtle.Turtle()
3  tori.shape('turtle')
4
5  def ramble(t, len):
6      if len <= 10:
7          t.stamp()
8      else:
9          for i in range(3):
10             t.right(120)
11             t.forward(len)
12         ramble(t, len//2)
```

i. `ramble(tori,2)`



ii. `ramble(tori,80)`



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

(c) If you call `ramble(tori,2)`, which branches of the function are tested (check all that apply):

- ☐ The block of code at Line 7.
- ☐ The block of code at Lines 10-11.
- ☐ None of these blocks of code (lines 7, 10-11) are visited from this invocation (call).

(d) If you call `ramble(tori,80)`, which branches of the function are tested (check all that apply):

- ☐ The block of code at Line 7.
- ☐ The block of code at Lines 10-11.
- ☐ None of these blocks of code (lines 7, 10-11) are visited from this invocation (call).

5. Design an algorithm that takes a string and returns the most common vowel (that is: 'a','i','e','o', or 'u') in the string. If there are multiple vowels that occurs most often, return the first one alphabetically. Your algorithm, if given the input:

   ```
   "One Fish Two Fish.  -Dr. Seuss"
   ```

   would return **e** since of the three vowels that occur most often ('e','i','o'), 'e' 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

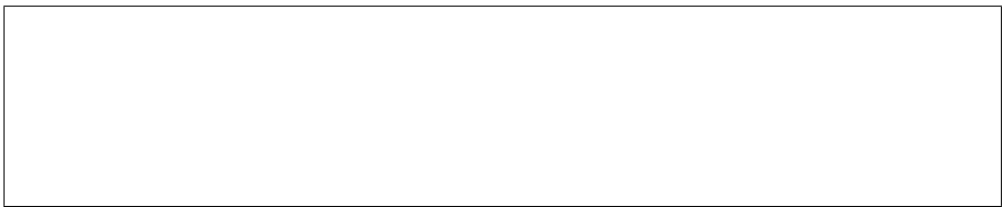
Process (as a concise and precise LIST OF STEPS / pseudocode):

(Assume libraries have already been imported.)

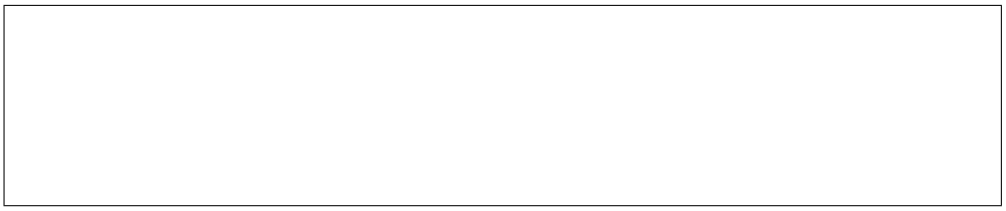
6. Fill in the following functions that are part of a program that draws with turtles:

   - `getData()`: gets the color and shape of a turtle and the number of sides of a polygon
   - `getTurtle()`: returns a turtle with color and shape
   - `drawPolygon()`: draws a polygon with n sides using turtle t

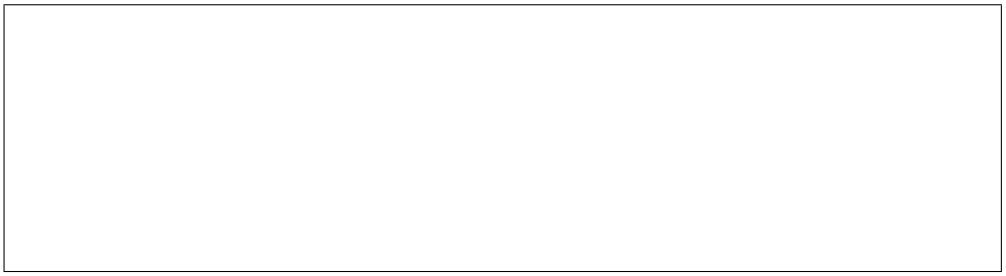
```
import turtle
def getData():
    """
    Asks the user for the color and shape of a turtle
    and the number of sides of a polygon.
    Returns the color and shape as strings and the sides as integer.
    """
```



```
def getTurtle(color, shape):
    """
    Returns a turtle with color and shape
    """
```



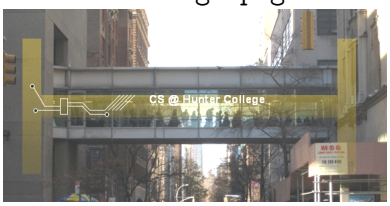
```
def drawPolygon(t, n):
    """
    Draws a polygon with n sides using turtle t

    """
```



7. Write a **complete Python program** that asks the user for a name of an image .png file and the name of an output file. The program then converts the image to grayscale, by setting the red and blue values for each pixel to its green value. It then saves the grayscale image to the specified output file. A sample run of your program should look like:

```
Enter name of the input file:  csBridge.png
Enter name of the output file:  gray_bridge.png
```

csBridge.png



gray_bridge.png

8. (a) Consider the following MIPS program:

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

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

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

(b) Consider the MIPS code:

```
1  ADDI $sp, $sp, -7
2  ADDI $t0, $zero, 110
3  ADDI $s2, $zero, 122
4  SETUP: SB $t0, 0($sp)
5  ADDI $sp, $sp, 1
6  ADDI $t0, $t0, 2
7  BEQ $t0, $s2, DONE
8  J SETUP
9  DONE: ADDI $t0, $zero, 0
10 SB $t0, 0($sp)
11 ADDI $sp, $sp, -6
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 whole message printed?	
iv) Detail the changes needed to the code to print first half of the message:	

9. (a) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Rage rage ";
    cout << "against\nthe dying ";
    cout << "of the light."<<endl<<"Dylan";
    cout << "Thomas";
    return 0;
}
```

Output:

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

```
#include <iostream>
using namespace std;
int main()
{
    [                                                    ]
    [                                                    ]
    while ( (myst < 15) && (quest > 0) )
    {
        cout << myst << "\t" << quest << endl;  myst += 5;
        quest--;
    }
    return 0;
}
```

Output:

```
-5   5
0    4
5    3
10   2
```

(c) What is the output:

```
#include <iostream>
using namespace std;
int main()
{
    for (int i = 1; i <= 5; i++)
    {
        for (int j = 0; j < 5; j++)
            if (i%2 == 0)
                cout << "4";
            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() {
    float temp = -50.0;
    while ((temp < -10) || (temp > 120))
    {
        cout << "Enter temperature: ";
        cin >> temp;
    }
    cout <<"Temp entered is " <<temp <<".\n";
    return 0;
}
```

Python program:

(b) Write a **complete C++ program** that asks for a positive whole number, `num`, and prints out the partial products up to `num!` (e.g. 1, 1×2, $1 \times 2 \times 3$,..., $1 \times 2 \times \cdots \times num$).

A sample run of your code:

```
Enter num:  5
1
2
6
24
120
```

SCRATCH PAPER

SCRATCH PAPER

CSCI 127 Reference Sheet, Fall 2025

Turtles: Let `t` be a turtle.

Function	Description
`t=Turtle.turtle()`	Creates turtle `t`.
`t.forward(x)`	Moves `t` forward `x` steps.
`t.backward(x)`	Moves `t` backward `x` steps.
`t.left(x)/t.right(x)`	Turns `t` left/right `x` degrees.
`t.penup()/t.pendown()`	Lifts `t`'s pen up/down.
`t.stamp()`	Stamps at `t`'s current location.
`t.goto(x,y)`	Moves `t` to `(x,y)`.

String Methods: Let `s` be a string.

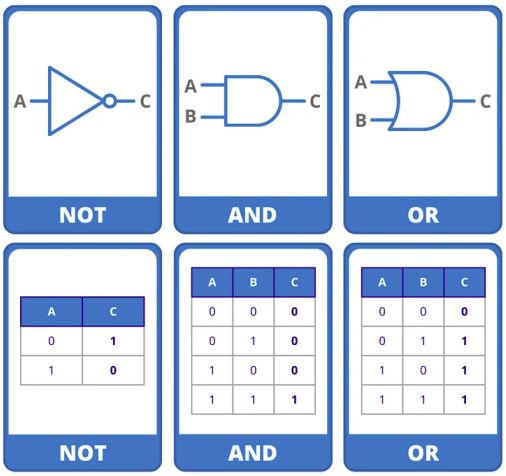
Function	Description
`len(s)`	Returns the length of `s`.
`s.lower()`	Returns `s` as lower case characters.
`s.upper()`	Returns `s` as upper case characters.
`s.count(t)`	Returns count of t in s.
`s.find(t)`	Returns index of t in s (-1 not found).
`s.split(d)`	Splits `s` into list of strings on `d`.
`s.join[lst]`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

Containers: Lists, Ranges & Dictionaries.

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

Functions:

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



(from truthtablegen.com)

numpy: Let `np` be the `numpy` package.

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

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

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

C++:

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

(From wikipedia commons)

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]