ASCII's Conversion to Binary

The ASCII conversion chart:

65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

ASCII		64	32	16	8	4	2	1
65	A	1	0	0	0	0	0	1

ASCII		64	32	16	8	4	2	1
66	B	1	0	0	0	0	1	0

ASCII		64	32	16	8	4	2	1
67	C	1	0	0	0	0	1	1

ASCII		64	32	16	8	4	2	1
97	a	1	1	0	0	0	0	1

ASCII		64	32	16	8	4	2	1
98	b	1	1	0	0	0	1	0

ASCII		64	32	16	8	4	2	1
98	c	1	1	0	0	0	1	1

Patterns in ASCII

ASCII		64	32	16	8	4	2	1
68	D	1	0	0	0	1	0	0
69	E	1	0	0	0	1	0	1
70	F	1	0	0	0	1	1	0

ASCII		64	32	16	8	4	2	1
100	d	1	1	0	0	1	0	0
101	e	1	1	0	0	1	0	1
102	f	1	1	0	0	1	1	0

64 bit
1 = letter
0 = no letter

32 bit
1 = lowercase
0 = uppercase

Remaining bits
Position in alphabet

ASCII is how letters are stored on computers. When you type in a phrase like: "Hello, world", it gets translated to ASCII and then to binary. The binary version is what is stored in the document that you save on the computer.

Letters	H	e	l	l	o	,		w	o	r	l	d
ASCII	72	101	108	108	111	44	32	119	111	114	108	100
Binary	1001000	1100101	1101100	1101100	1101111	0101100	0100000	1110111	1101111	1110010	1101100	1100100

ASCII normally adds an extra bit as a "check bit" for error testing. Thus, each letter requires 8 bits or one byte.

A String is an array of chars.

Strings have a lot of built in functions (that are handy)

```
String s = "apple";
```

0	1	2	3	4
a	p	p	l	e

What would be returned by each of these methods?

Fill in the memory diagram.

```
5        s.length()
APPLE    s.toUpperCase()
apple    s.toLowerCase()
0        s.indexOf('a')
p        s.charAt(2);
112      (int) (g.charAt(2))
bpple    s.replace('a','b')
error    s.substring(2,6)      2,4   pl
false    s.equals("bob")       1,3   pp
false    s.compareTo("bob")>0
```

apple > bob

Write the method signature for each method.

```
s.length()
```
public int length()

```
s.toUpperCase()
```
public String toUpperCase()

```
s.toLowerCase()
```
public String toLowerCase()

```
s.indexOf('a')
```
public int indexOf(char c)

```
s.charAt(2)
```
public char charAt(int i)

Write the method signature for each method.

```
s.replace('a','b')
```
public String replace(char c1, char c2)

```
s.substring(2,6)
```
public String substring(int start, int end)

```
s.equals("bob")
```
public boolean equals(String s)

```
s.compareTo("bob")>0
```
public int compareTo(String s)

Assume you only have to deal with this name.

```
String n = "Susan Gorski";
```
0 1 2 3 4 5 6 7 8 9 10 11

```
s.length()
s.toUpperCase()
s.toLowerCase()
s.indexOf('a')
s.charAt(2);
(int) (g.charAt(2))
s.replace('a','b')
s.substring(2,6)
s.equals("bob")
s.compareTo("bob")>0
```

Print out the student's first name.

```
String first=n.substring(0,5);
```

Print out the student's last name.

```
String last =n.substring(6, n.length());
```
12

Print out the student's initials.

```
char first =n.charAt(0);
char last =n.charAt(6);
```

First, understand the input & output:

Enter a phone number:

p =

0	1	2	3	4	5	6	7	8	9
5	1	9	3	9	6	1	2	3	3

The name number, reformatted:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
(	5	1	9	)		3	9	6	-	1	2	3	3

To build up the new number:

```
String pn = "(" + p.substring(0,3) + ")  " +
            p.substring(3,6) + "-" +
            p.substring(6,10);

result.setText(pn);
```

Assume ANY length of name, in this format.

```
String n = "Manco Capac";
String n = "Katakura Michinao";
```

```
s.length()
s.toUpperCase()
s.toLowerCase()
s.indexOf('a')
s.charAt(2);
(int) (g.charAt(2))
s.replace('a','b')
s.substring(2,6)
s.equals("bob")
s.compareTo("bob")>0
```

Print out the student's first name.

```
String first=n.substring(0,n.indexOf(" "));
```

Print out the student's last name.

```
String last=n.substring(n.indexOf(" ")+1,n.length());
```

Print out the student's initials.

```
char f = n.charAt(0);
char l = n.charAt(n.length(" ")+1);
```

```
String y = "Yourba";
```

```
s.length()
s.toUpperCase()
s.toLowerCase()
s.indexOf('a')
s.charAt(2);
(int) (g.charAt(2))
s.replace('a','b')
s.substring(2,6)
s.equals("bob")
s.compareTo("bob")>0
```

Divide any even length word in half.

```
if(y.length()%2 ==0){
        String f = y.substring(0, y.length()/2);
        String l = y.substring(y.length()/2, y.length());
        result.setText(f + " " + l);
}
```

```
String b = "Brahmagupta";
```

```
s.length()
s.toUpperCase()
s.toLowerCase()
s.indexOf('a')
s.charAt(2);
(int) (g.charAt(2))
s.replace('a','b')
s.substring(2,6)
s.equals("bob")
s.compareTo("bob")>0
```

Make hangman dashes for this word. Put a space between each dash.

```
String clue = "";
for (int i=0; i<b.length(); i++) {
    clue += " _ ";
}
```

```
String c = "Chinchorro mummies";
```

Print out the ASCII value of each letter, with a space between them.

```
String a = "";
for(int i=0; i < c.length(); i++){
    a += (int) c.charAt(i);
    a += " ";
}

result.setText(a);
```

```
s.length()
s.toUpperCase()
s.toLowerCase()
s.indexOf('a')
s.charAt(2);
(int) (g.charAt(2))
s.replace('a','b')
s.substring(2,6)
s.equals("bob")
s.compareTo("bob")>0
```