Introduction to Computers and Programming LAB-Midterm 2014/11/12 Time: 2.5 hrs

※Please create a new folder. Name the folder as: Student ID-Name (XXXXXXX-○○○). Inside the folder, your file format will be Q_1.c, Q_2.c, etc. There will ONLY be a total of 5 .c files in your folder (wrong file name or format will cause score deductions).

※No Internet. No discussions.

※In Problem 1~5, please wrap each of your code inside main(){} with while(1){}

※Using Array is not allowed

※The class is for C language (C89), so do not use C++, If your program cannot be compiled, you will get zero score for the question.

※Before you use any variables, make sure you have assigned values to them. Some IDE's, such as Dev-C++, may not automatically initialize the variables.

※Your programs will be checked (by a tool) for the programming integrity. Be honest with your own works.

※Your output must comply with the sample output format.

1. (10%) Largest and smallest number

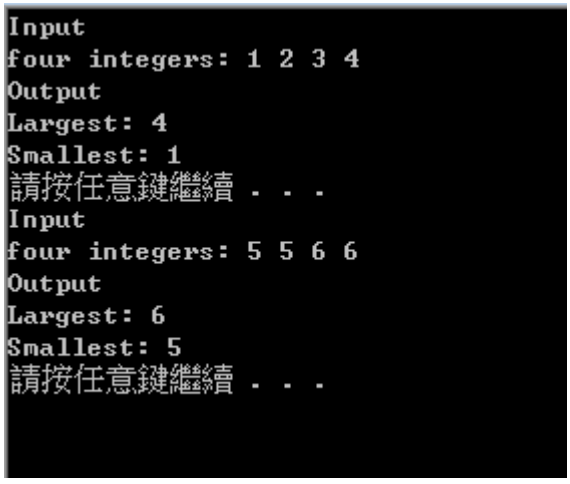
   Write a program that finds the largest and smallest of four integers entered by the user:

   Enter four integers: 21 43 10 35

   Largest: 43

   Smallest: 10

   Hint: four *if* statements are sufficient, but you can use more.



2. (15%) Robot

   There is an ongoing robot designed to search the ground of Mars. It can walk 4 different directions including North: 0, East: 1, South: 2 and West: 3, which are numbered from 0 to 3. At the beginning, researchers will put it at the start point with x, y coordinate. Then give it N instructions, and each instruction includes one direction code, C, and one distance, D, meaning that robot should walk D meters in C direction. So the robot will do N times of moving.

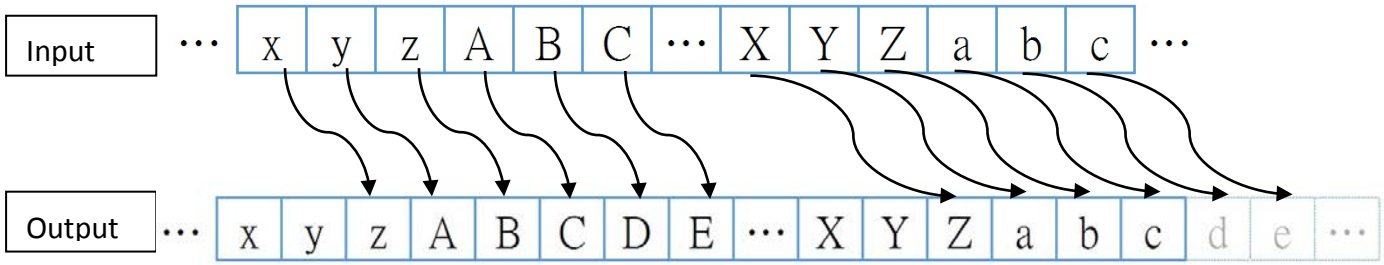
   Finally, you should give us the x', y' coordinates of the destination.

```
Input
the coordinates of the robot: 0 0
the number of instructions, N : 2
0 2
1 2
Output
Now the robot is at (2, 2)
請按任意鍵繼續 . . .
Input
the coordinates of the robot: 5 5
the number of instructions, N : 3
2 3
3 3
0 0
Output
Now the robot is at (2, 2)
請按任意鍵繼續 . . .
```

3. (15%) Caesar cipher(加密)

In cryptography, a Caesar cipher, also known as Caesar's cipher, the shiftcipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet. For example, with a left shift of 3, D would be replaced by A, E would become B, and so on. The method is named after Julius Caesar, who used it in his private correspondence – *from wiki*

Today TAs suggest a new way to cipher both A-Z and a-z. That is to connect A-Z sequence with a-z sequence, so the length of the sequence becomes 52 and we use **right shift (位移) of 2**, so we can handle the sentence with both uppercase and lowercase letters.



Write a program to implement TA's cipher, input a sentence and then cipher them, note that you should only process uppercase letters and lowercase letters.

```
Input a sentence: Mtcp Kw BcYb zmbw
Over My Dead Body
請按任意鍵繼續 . . .
Input a sentence: G Jmtc NpmepYkkgle
I Love Programming
請按任意鍵繼續 . . .
```

ASCII code table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

4. (15%) Decoding

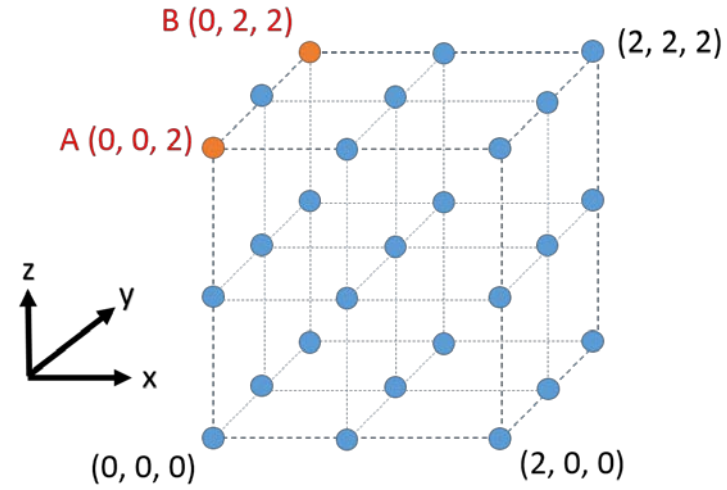
Write a program to decode the input bit patterns to decimal, using the floating-point format in which the left-most bit is the sign bit, followed by 3 bits for exponent, and the rest for mantissa. Note that exponent is denoted in excess notation.

```
Input
bit patterns: 01011010
Output
decimal number in float type: 1.250000
請按任意鍵繼續 . . .
Input
bit patterns: 10111001
Output
decimal number in float type: -0.281250
請按任意鍵繼續 . . .
```

5. (15%) Clustering

Write a program to compute the numbers of integer points that are closest to 2 points in a cube which is ranged by two opposite corners, (0, 0, 0) and (S, S, S) in 3-dimensional space. For example, there are 27 integer points in the cube with two corners, (0, 0, 0) and (2, 2, 2) showing in below graph. Now given 2 center points, A (0, 0, 2) and B (0, 2, 2) within the cube. We want to compute the number of points, except A and B, which are closet to these two center points respectively. For example, if we consider the point (2, 2, 2) then it is closest to B (0, 2, 2). But, if the distances of the point to A and B are the same, then we choose the center point that has the smaller x coordinate (and if still a tie, choose smaller y, and finally compare z coordinate). For instance, (1, 1, 1) is closest to both A (0, 0, 2) and B (0, 2, 2). However, A (0, 0, 2) has the same x coordinate with B but the smaller y coordinate than B, so (1, 1, 1) belongs to A point.

Hint: Formula of distance d of two point u (u_x, u_y, u_z), v (v_x, v_y, v_z):
$d^2 = (u_x - v_x)^2 + (u_y - v_y)^2 + (u_z - v_z)^2$




```
Input
the size S of cube: 2
Point A: 0 0 2
Point B: 0 2 2
Output
number of cluster A = 17
number of cluster B = 8
請按任意鍵繼續 . . .
Input
the size S of cube: 5
Point A: 1 2 3
Point B: 5 4 1
Output
number of cluster A = 151
number of cluster B = 63
請按任意鍵繼續 . . .
```