CS3335 Assignment 3 Due:09/7/17

1. (6 points) This question is designed to test your understanding of pointer arithmetic. Suppose SIZE is symbolic constant with value 100 (`#define SIZE 100`). If the declaration is

```
char a[SIZE], *p=a;
int i;
```

(Notice that I am deliberately using an array of `chars` because each `char` is stored in one byte), I want to fill the array in a very simple way.

```
for (i = 0; i < SIZE; i++)
   a[i] = i;
```

What is printed? Well, the answer is actually system-dependent. Here you are asked to put the code above (and some additional code, if necessary) into a piece of C program, run it on your computer, and **fill up** the following chart with outputs your program produces:

Code	Output
`sizeof(char)`	1
`sizeof(int)`	4
`sizeof(double)`	8
`sizeof(long double)`	16
`printf( "%d\n", *(p+3) );`	3
`printf( "%d\n", *(char *)((int *)p+3) );`	12
`printf( "%d\n", *(char *)((double *)p+3) );`	24
`printf( "%d\n", *(char *)((long double *)p+3) );`	48

Next, imagine your program is being executed on a different system that has `char`, `int`, `double`, and `long double` of different sizes as shown in the following chart. **Fill up** the rest of the chart.

Code	Output
`sizeof(char)`	1
`sizeof(int)`	2
`sizeof(double)`	16
`sizeof(long double)`	32
`printf( "%d\n", *(p+3) );`	3
`printf( "%d\n", *(char *)((int *)p+3) );`	6
`printf( "%d\n", *(char *)((double *)p+3) );`	48
`printf( "%d\n", *(char *)((long double *)p+3) );`	96