

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;
```

Now consider

```
printf( "%d\n", *(p+3) );
printf( "%d\n", *(char *)((int *)p+3) );
printf( "%d\n", *(char *)((double *)p+3) );
printf( "%d\n", *(char *)((long double *)p+3) );
```

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
<code>sizeof(char)</code>	
<code>sizeof(int)</code>	
<code>sizeof(double)</code>	
<code>sizeof(long double)</code>	
<code>printf("%d\n", *(p+3));</code>	
<code>printf("%d\n", *(char *)((int *)p+3));</code>	
<code>printf("%d\n", *(char *)((double *)p+3));</code>	
<code>printf("%d\n", *(char *)((long double *)p+3));</code>	

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
------	--------

<code>sizeof(char)</code>	1
<code>sizeof(int)</code>	2
<code>sizeof(double)</code>	16
<code>sizeof(long double)</code>	32
<code>printf("%d\n", *(p+3));</code>	
<code>printf("%d\n", *(char *)((int *)p+3));</code>	
<code>printf("%d\n", *(char *)((double *)p+3));</code>	
<code>printf("%d\n", *(char *)((long double *)p+3));</code>	

2. (17 points) A stack is a data structure in which only the top element can be accessed in the FILO fashion.

You job is to complete the skeleton program `stack.c` in attachment such that basic operations like `push()` and `pop()` can be carried out on the stack created in `main()`. Finish `main()` to test your functions. See detailed instructions in `stack.c`.

What to turn in?

Create a tarball file by the name of `cs3335_a3_yourlastname.tar` that includes

- A PDF file by the name of `a3q1.pdf` that contains your answer to question 1.
- The completed source code file `stack.c` for question 2.

Submit the tar ball file through BlazeVIEW by the due time.