202-A01 Tracing programs which are using pointers

Work to do

You can do this exercise purely on paper, using and filling the table provided to you below. Make sure that you also note if a given line would cause a bug and explain its nature.

It is also a good idea to type this code and compile it then add to it a bunch of printf statements to double check that your trace is actually correct. When you do so, you will come up with questions (e.g. “I though it worked this way, why is the printf telling me otherwise) which you can post on the course’s forums to get help with.

If you do not verify your trace by implementing the code you will probably leave errors in it which you will commit again at our next exam. Warn be thee.

Here is the program you will have to trace;

```
#include <stdio.h>
int main ()
{
1.      int array[5] = { 0,1,2,3,4} ;
2.      int * p1, **p2 ;
3.      int * parray[2] = { NULL , NULL };
4.      p1= & array[2] ;
5.      *p1 = 20 ;
6.      p2 = & p1 ;
7.      ** p2 = 30 ;
8.      *p2 = NULL;
9.      parray[0] = & array[0];
10.     parray[1] = & **p2;
11.     *parray[0] = 30 ;
12.     *parray[1] = 300;
}
```

Use the table on next page to trace its execution line per line by hand.

Testing

Here are the variables you will have to trace and the memory addresses we are going to assume they are located at. These will come handy when you assign the address of a variable to a pointer. You will be able to keep track of who points to who by filling in these memory addresses as contents of your pointer variables.

	Variables									
	array					p1	p2	parray		
Memory Address	1000	1004	1008	1012	1016	1020	1024	1028	1032	

The following table is when you will be tracing how values of each variable will change as the program executes

Value after executing Line #	Variables values													
	array					p1		p2			parray		*parray	
	[0]	[1]	[2]	[3]	[4]	p1	*p1	p2	*p2	**p2	[0]	[1]	[0]	[1]
1														
2														
3														
4														
5														
6														
7														
8														
9														
10														
11														
12														