1. Consider the correct version of the function `swap()` that we discussed in class (and is in the text). Write a similar function `rotate()` that accepts four arguments, each of type pointer to integer, and “rotates” the values that they point to, setting the first `int` to the second `int`’s original value, the second to third, etc.,. For example, after executing the following code in another function:

```c
int i1, i2, i3, i4;
i1 = 1; i2 = 2; i3 = 3; i4 = 4;
rotate (&i1, &i2, &i3, &i4);
```

- we should have `i1` set to 2, `i2` set to 3, `i3` set to 4, `i4` set to 1.

The constraint is that in the body of the function `rotate()`, you can only define new variables of type `int*` (and use them appropriately as pointers). You may not define any other type of variables, and in the function code you may not use any global variables, or call any other function.

2. Consider the following complete program (intended to be treated as a single file and compiled into a binary executable). In the table below, write down the values of the variables specified after the corresponding line of code has been executed in the spaces provided. (If it is not possible to write down the value, indicate why.) For any question, interpret the variable name the same way that the compiler would interpret it at that point in the code, according to the rules of scope, linkage, etc. For ease of reference, some lines of code have been numbered, these numbers are not part of the actual code. The function `main()` is obviously executed only once. The function `ipow()` is called three times by `main()`, so the question specifies which call of `ipow()` is being referred to when specifying variable values after lines of code inside that function.

```c
#include <stdio.h>
#include <stdlib.h>
01. int iG = 5;
02. int jG;
03. unsigned int kuG;
04.
05. float ipow (int *p1, int *p2)
06. {
07.     int i;
08.     float result = 1;
09.    10.     for (i=0; i<*p2; i++) {
11.         result = result * (*p1);
12.     }
13.     return result;
14. }
```
12. }
13. 
14. return (result);
15. }
16. 
17. int main (void)
18. {
19.    int i, j, k, *pa, *pb;
20. 
21.    i = -2; pa = &k;
22.    kuG = i;
23. 
24.    k = ipow (&i, &iG);
25.    k = ipow (&i, &jG);
26.    k = ipow (&kuG, &kuG);
27. 
28.    i = j;
29. }

<table>
<thead>
<tr>
<th>After line 24 function call has returned and before 25 begins</th>
<th>i</th>
<th>iG</th>
<th>k</th>
</tr>
</thead>
<tbody>
<tr>
<td>After line 25 function call has returned and before 26 begins</td>
<td>i</td>
<td>jG</td>
<td>k</td>
</tr>
<tr>
<td>After line 26 call has returned and before line 28 begins executing</td>
<td>i</td>
<td>jG</td>
<td>k</td>
</tr>
<tr>
<td>After line 12 has finished executing, during first call of ipow ()</td>
<td>i</td>
<td>iG</td>
<td>jG</td>
</tr>
<tr>
<td>After line 8 has finished executing, during third call of ipow ()</td>
<td>i</td>
<td>*pa</td>
<td>*p1</td>
</tr>
</tbody>
</table>