David Montenegro

Assignment

1. Define a function that takes a character array ca as an argument. Inside the function, print out the values of &ca and &(ca[0]) and &(ca[1]).
   &ca = 0x7fff4995b528
   &(ca[0]) = 0x7fff4995b543
   &(ca[1]) = 0x7fff4995b544

2. Define another function that takes a character pointer pa as an argument. Inside the function, print out the values of &pa and &(pa[0]) and &(pa[1]) and ++pa.
   &pa = 0x7ffccefc0948
   &(pa[0]) = 0x7ffccefc0973
   &(pa[1]) = 0x7ffccefc0974
   ++pa = 0x7ffccefc0974

3. Set up a global character array ga and initialize it with the letters of the alphabet. Call the two functions using this global as the parameter. Compare the values that you print out.
   &ca = 0x7fff91ab2b88
   &(ca[0]) = 0x561ec30a3010
   &(ca[1]) = 0x561ec30a3011
   &pa = 0x7fff91ab2b88
   &(pa[0]) = 0x561ec30a3010
   &(pa[1]) = 0x561ec30a3011
   ++pa = 0x561ec30a3011

   &ca and &pa are the same.
   &(ca[0]) and &(pa[0]) are the same.
   &(ca[1]), &(pa[1]), and ++pa are the same.

4. In the main routine, print out the values of &ga and &(ga[0]) and &(ga[1]).
   See problem 5.

5. Before running your program, write down which values you expect to match, and why. Account for any discrepancies between your expected answers and observed results.
   &ga will differ from &ca and &pa because &ca and &pa are local to the functions they are located in.
   &ga and &(ga[0]) will match &(ca[0]) and &(pa[0]) because the contents of the arrays are the same.
   &(ga[1]) will match &(ca[1]), &(pa[1]) and ++pa because the contents of the arrays are the same.
   &ca = 0x7fff1015b928
   &(ca[0]) = 0x557f4f740010

```
&(ca[1]) = 0x557f4f740011
&pa = 0x7fff1015b928
&(pa[0]) = 0x557f4f740010
&(pa[1]) = 0x557f4f740011
++pa = 0x557f4f740011
&ga = 0x557f4f740010
&(ga[0]) = 0x557f4f740010
&(ga[1]) = 0x557f4f740011
```