Oh boy! We get to do some actual real computer science. When writing code programmers tend to need to sort lists very frequently. Because of this, doing a really fast sort is incredibly important for basically any project. Normally we would use pre-written code to do this for us (using a sort method called quicksort), but I want you guys to write a simple sorting algorithm to practice on. The sorting method we will use is called Insertion Sort.

You can google insertion sort to get an explanation of how it works (Geeksforgeeks is a good website), but I'll talk about it a bit here. Insertion sort if often compared to how you sort playing cards in your hands. You start from the left, and every time you find something that isn't in order you move it to its correct place. You mentally have a "sorted" and "unsorted" section of your cards, so everything on the left is sorted, and you move cards one at a time from sorted to unsorted. When you get to the end of your hand every card should be sorted!

So, let's discuss. I'll start with a paragraph on setting up your code, which you should read. Then, I'll put in a paragraph on some simple methods, and finally I'll have a paragraph with some actual pointers on what your code should do. I suggest starting after reading the first (and maybe second) paragraph, and then moving on to the other paragraph as you get stuck.

Your sorting array needs to be universal, so it needs to take any size of array. Ideally you should have a variable that controls the size of the array so that you can change it up. One way to do this uses `#define` to create a constant variable. You cannot put a variable into a new array to say how big it needs to be (don't ask), but you can use a define to create a constant. At the top of your file put a line that says something like

```
#define name 10
```

This tells your compiler to replace every single place that says "name" with a 10 before starting compilation, so it doesn't count as a variable. You can then use to create a new random array of that size using code like this:

```
#define size 10
int list[size];
srand(time(NULL));
int i = 0;
while(i < size){
      list[i] = rand();
      i++;
}
```

You will then have an array with 10 random numbers in it that you can try to sort. Remember, since you are using arrays, you will need to manually check for array limits so you don't go over. Also make sure that you save your code frequently, and have a cout loop at the end so you can check the results.

Okay, this start of paragraph 2. If you want to try this all on your own, don't keep reading!

So, let's go back to the example of sorting playing cards in your hand by number. You need to sort every card, so you will have a loop that starts at 0 and goes up to `size`. But for each card you need to work backwards and keep checking if the current card is greater, so you need another loop inside the first loop that goes from card `i` and works down to zero. You can stop once you find a card that is smaller than the current card, because every card to the left of the current card should be sorted, which means that once you find a spot between a larger number and a smaller number, you're done. (That's a bit of a messy explanation, you may want to go online for some visual examples). Finally, you can choose to either move the current card one slot at a time, or all at once. If you do it the second way, don't forget that you need make space for the new card, so to move card #5 into space #2 you'll need to put #5 into a variable, and then move 4->5, 3->4, 2->3, and then finally you can put 5 into 2. This is going to be ANOTHER loop, so it might be a bit messy.

Okay, paragraph 3! DON'T READ THIS NEXT SECTION UNTIL YOU HAVE AT LEAST TRIED TO DO SOMETHING!!!!

So, since we have (at least) three loops, we're going to need two iterator variables. I suggest `i` and `j` for the first two, but use a more helpful name for any extra ones (like `shiftcounter` or something). Your loops should look something like this:

```
int i = 0;
while(i < size){
      int j = i-1;
      while(j >= 0){
            if(array[i] < array[j]){
                  int hold = array[i];
                  array[i] = array[j];
                  array[j] = hold;
            }
            j--;
      }
      i++;
}
```