Homework #6

Due date: 12/15

Part A – Gnome sort

Let a is a global array declared by

```
const int sz=30;
int a[sz];
```



Write the function

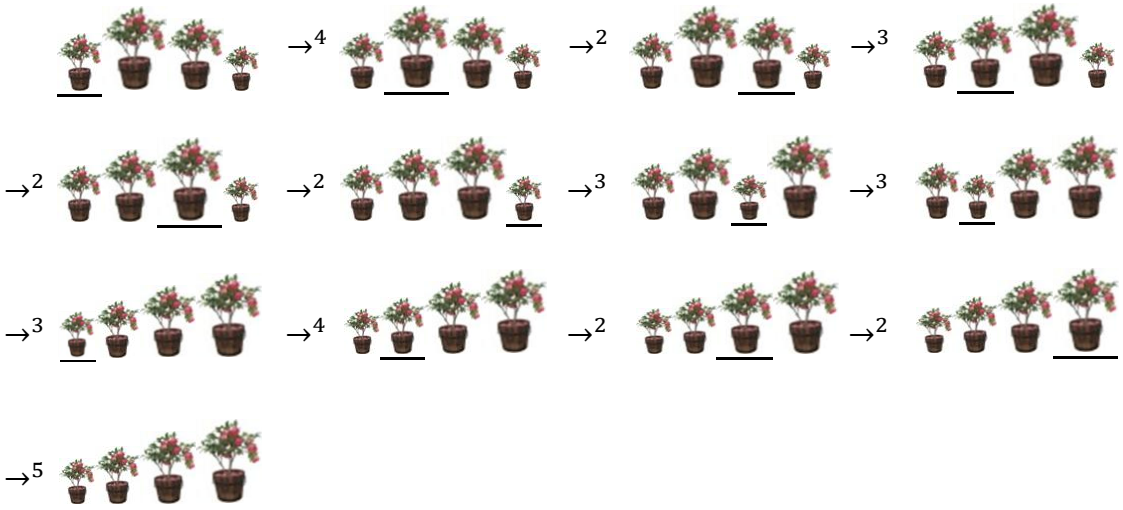
```
void gnomesort(int n);
```

to sort the elements in $a[0..n-1], n \leq sz$, into nondecreasing order by gnome sort, as described below.

Gnome Sort is based on the technique used by a garden gnome to sort a line of flower pots. Imagining that the array elements are flower pots, the gnome sort goes through the following steps.

1 Basically, the garden gnome scans the flowerpots from $a[0]$ to $a[n-1]$, looking at the current flowerpot $a[i]$ and the previous flowerpot $a[i-1]$.
2 If they are in right order, the garden gnome steps one flowerpot forward to $a[i+1]$.
3 If they are in wrong order, the garden gnome first swaps the two flowerpots $a[i]$ and $a[i-1]$ and then steps one flowerpot backward to $a[i-1]$.
4 If there is no previous flowerpot, the garden gnome simply steps one flowerpot forward.
5 If there are no more flowerpots to consider, stop.
The garden gnome is done.

For example, let $a[0..3] = 1,3,2,0 =$ 
The gnome sort proceeds as follows, where each underlined flowerpot is the flowerpot under consideration and $\rightarrow^k$ indicates that the kth step is applied.



In terms of numbers, the steps are:

$$\underline{1},3,2,0 \rightarrow^4 1,\underline{3},2,0 \rightarrow^2 1,3,\underline{2},0 \rightarrow^3 1,\underline{2},3,0 \rightarrow^2 1,2,\underline{3},0 \rightarrow^2 1,2,3,\underline{0}$$
$$\rightarrow^3 1,2,\underline{0},3 \rightarrow^3 1,\underline{0},2,3 \rightarrow^3 \underline{0},1,2,3 \rightarrow^4 0,\underline{1},2,3 \rightarrow^2 0,1,\underline{2},3 \rightarrow^2 0,1,2,\underline{3}$$
$$\rightarrow^5 0,1,2,3$$

In fact, gnome sort is similar to insertion sort; but they differ in two respects. First, it moves an element to its proper place by a series of swaps, as in bubble sort. Second, after stepping backward to move an element from its original place to its proper place, it steps forward to look for the next element to insert (the garden gnome is too old to remember the element's original place and can't start from there to look for the next element).

Part B – Cocktail sort

Write the function

```
void cocktailsort(int n);
```

to sort the elements in $a[0..n-1]$ into nondecreasing order by cocktail sort, as described below.

Cocktail sort (or bidirectional bubble sort) is a variation of bubble sort that sorts in both directions on each pass through the sequence of elements. That is, each pass has two phases: phase A moves the largest element down and phase B moves the smallest element up.

For example, let $a[0..5] = 6,5,4,3,2,1.$

The cocktail sort proceeds as follows, where 1A denotes phase A of pass 1, 1B denotes phase B of pass 1, and so on. The red-colored numbers are elements of sorted subarrays

6		5		1		1		1		1		1
5		4		5		4		2		2		2
4	1A →	3	1B →	4	2A →	3	2B →	4	3A →	3	3B →	3
3		2		3		2		3		4		4
2		1		2		5		5		5		5
1		6		6		6		6		6		6

Your function shall terminate whenever no swap occurs in a phase:

7		5		1		1		1			
5		3		5		3		2		2	
3		1		3		2		3		3	
1	1A →	6	1B →	2	2A →	5	2B →	4	3A →	4	finish, no swap in phase 3A
6		4		6		4		5		5	
4		2		4		6		6		6	
2		7		7		7		7		7	

Furthermore, you function shall extend the sorted subarray as far as possible on each phase:

1		1		1		1		1	
7		5		2		2		2	
5		3		5		3		3	
3		4		3		4		4	
4	1A →	6	1B →	4	2A →	5	2B →	5	finish, no swap in phase 2B
6		2		6		6		6	
2		7		7		7		7	
8		8		8		8		8	
9		9		9		9		9	

Sample test

Suffice it to run the function **main** below, where the function **in** uses **rand()%10** to generate test data.

```
int main(void)
{
    // The garden gnome figurines originate in the 19th century (1801~1900) Germany.
    srand(180121900);
    for (int i=1;i<=8;i++) {
        in(sz);
        printf("Before sorting ...\n");
        out(sz);
        switch (rand()%2) {
        case 0: gnomesort(sz); printf("Sorted by gnome sort\n"); break;
        case 1: cocktailsort(sz); printf("Sorted by cocktail sort\n");
        }
        out(sz);
        printf("\n");
    }
}
```

Sample run

```
Before sorting ...
2 7 3 0 7 5 2 1 0 6 7 8 5 4 5 0 9 0 9 8 5 9 0 0 8 7 7 5 9 5
Sorted by cocktail sort
0 0 0 0 0 0 1 2 2 3 4 5 5 5 5 5 5 6 7 7 7 7 7 8 8 8 9 9 9 9

Before sorting ...
9 5 0 1 1 6 7 2 9 1 4 3 9 5 3 7 9 5 2 3 3 9 5 6 2 5 5 5 2 4
Sorted by cocktail sort
0 1 1 1 2 2 2 2 3 3 3 3 4 4 5 5 5 5 5 5 5 6 6 7 7 9 9 9 9 9

Before sorting ...
9 2 4 5 3 0 7 6 8 6 3 6 9 3 2 6 2 2 6 4 2 0 5 9 6 0 0 7 4 2
Sorted by gnome sort
0 0 0 0 2 2 2 2 2 2 3 3 3 4 4 4 5 5 6 6 6 6 6 6 7 7 8 9 9 9
```

```
Before sorting ...
5 5 4 8 6 9 5 0 3 9 9 4 2 2 5 1 5 1 5 0 1 8 5 5 0 9 7 5 3 1
Sorted by gnome sort
0 0 0 1 1 1 1 2 2 3 3 4 4 5 5 5 5 5 5 5 5 5 6 7 8 8 9 9 9 9

Before sorting ...
1 9 5 3 9 8 4 8 8 6 2 1 5 0 2 5 8 7 6 6 4 9 6 5 9 2 4 0 4 1
Sorted by cocktail sort
0 0 1 1 1 2 2 2 3 4 4 4 4 5 5 5 5 6 6 6 6 7 8 8 8 8 9 9 9 9

Before sorting ...
5 1 5 6 0 6 8 2 4 3 3 4 6 8 1 6 0 9 0 6 9 5 6 9 5 5 3 8 6 1
Sorted by gnome sort
0 0 0 1 1 1 2 3 3 3 4 4 5 5 5 5 5 6 6 6 6 6 6 6 8 8 8 9 9 9

Before sorting ...
1 9 6 0 7 2 2 2 6 8 4 6 4 2 3 4 2 5 1 3 1 2 1 3 3 5 7 8 8 9
Sorted by gnome sort
0 1 1 1 1 2 2 2 2 2 2 3 3 3 3 4 4 4 5 5 6 6 6 7 7 8 8 8 9 9

Before sorting ...
5 4 1 1 3 4 5 3 0 8 5 1 9 6 2 0 2 5 8 7 0 1 7 7 6 7 4 2 7 1
Sorted by cocktail sort
0 0 0 1 1 1 1 1 2 2 2 3 3 4 4 4 5 5 5 5 6 6 7 7 7 7 7 8 8 9
```