Lab Test Week 10 : Sorted Lists

Examination Rules

This is an exam. Please:

- Wait to be seated by a Lab. supervisor.
- Do not talk to anyone else.
- Do not use a Web browser, unless instructed to do so (e.g. to submit files to SAFE).
- Do not use an email connection.
- Do not look in other peoples directories.
- Do not use code written in 'groups' with others students.
- **Do** use text books, lecture notes, man pages etc.

If you do not obey the above rules you will be asked to leave.
People arriving late may not be allowed into the Lab.
You have one hour. There is additional time in case a server crashes or some other problem occurs.

Submitting Your Work

As you complete each part, submit it online using the SAFE submission system.
If you complete Part 1, submit a file called `partone.c`.
If you complete Part 2, submit a file called `parttwo.c`.

Let a lab supervisor know how many parts you believe you've completed.
No partial marks are available for each part. The only possible scores for this exam are 0%, 60%, or 100%.

Code Style

- Your code must compile without warnings using the gcc flags:

  ```
  -pedantic -Wall -Wextra -Wfloat-equal -ansi -O2
  ```

- Do **NOT** use any global variables.
- If you are using `rand()` in your code, feel free to set the seed of the random number generator to give different outputs each time your program is run, for instance using :

```
srand(time(NULL));
```

remembering to `#include <time.h>`

Part 1 (60%)

Write a function that, when passed an integer array of size 4, fills it with random integers between 0 and 9 inclusive. If you were to print it out, such an array might look like:

```
3 0 9 3
```

Very occasionally when you run this function, the numbers produced would be in ascending order [1] . The following arrays are in ascending order:

```
4 6 6 8
2 2 2 2
6 7 8 9
```

The following arrays are not in ascending order:

```
1 0 7 9
1 4 0 9
2 9 3 4
```

Write a program that generates 1000 such lists and prints out whether or not they are in ascending order. Running the program may produce something like this for the first few lines :

```
8 1 2 9 Not Ascending
3 9 0 8 Not Ascending
8 5 0 9 Not Ascending
6 3 8 5 Not Ascending
6 1 1 5 Not Ascending
9 8 4 8 Not Ascending
1 0 3 0 Not Ascending
4 4 4 4 Ascending
7 6 3 1 Not Ascending
7 5 9 6 Not Ascending
2 1 7 8 Not Ascending
5 7 4 1 Not Ascending
8 5 9 7 Not Ascending
5 3 8 8 Not Ascending
3 1 8 9 Not Ascending
6 4 3 3 Not Ascending
3 8 6 0 Not Ascending
4 8 8 8 Ascending
```

[1] Ascending order means that the smallest numbers appear at the beginning of the list and that for any adjacent pair of integers $a[i] <= a[i+1]$.

```
9 7 7 6 Not Ascending
4 3 0 3 Not Ascending
0 9 2 5 Not Ascending
4 0 5 9 Not Ascending
```

Part 2 (40%)

Adapt the above code, so that lists of length 15 rather than 4 are generated, and they are filled with random integers in the range from 0 to 199. For each of these lists, find out whether any of the numbers occur twice (or more). For instance, in the list:

```
20 123 117 121 70 146 143 133 140 125 140 3 166 60 120
```

the number 140 appears twice.
When you run your code, the first few lines should look something like:

```
183  86 177 115 193 135 186  92  49  21 162  27  90  59 163
126 140  26 172 136  11 168 167  29 182 130  62 123  67 135
129   2  22  58  69 167 193  56  11  42  29 173  21 119 184
137 198 124 115 170  13 126  91 180 156  73  62 170 196  81 (170 appears 2 times)
105 125  84 127 136 105  46 129 113  57 124  95 182 145  14 (105 appears 2 times)
167  34 164  43 150  87   8  76 178 188 184   3  51 154 199
132  60  76 168 139  12  26 186  94 139 195 170  34 178  67 (139 appears 2 times)
  1  97 102 117  92  52 156 101  80  86  41  65  89  44  19
 40 129  31 117  97 171  81  75 109 127 167  56  97 153 186 (97 appears 2 times)
165 106  83  19  24 128  71 132  29 103  19  70 168 108 115 (19 appears 2 times)
140 149 196 123  18  45  46  51 121 155 179  88 164  28  41
150 193 100  34 164 124 114 187  56 143  91  27 165  59 136
 32 151  37  28  75   7  74 121  58 195  29  37  35 193  18 (37 appears 2 times)
 28 143  11 128 129 176   4  43 163  13 138   6  40 104  18
128  88 169 117 117 196 124 143  70 183  90  99 172 125  44 (117 appears 2 times)
190 105 139 154 186  69  82 142  64 197 107 155   4 148  11
 22  28  99 143 146 168 140  22 111  10   5   1  61 130  78 (22 appears 2 times)
```