SCC110: Software Development Term 1. C Programming.

Problem Set 4: Array Fun

The Goal
This week's lab aims to:

- The aim for today is to practice and consolidate what we've learnt so far and get to grips with how lists (arrays) work in C.
- You'll also get to practice iterating through arrays using loops
- There is a main problem with a number of sub-tasks of increasing difficulty. There is also a more advanced 'hacker edition' if you fancy more of a challenge!

Problem 1: Working with Arrays of Integers
You'll need to create a program that allows the user to enter 10 numbers into an integer array.

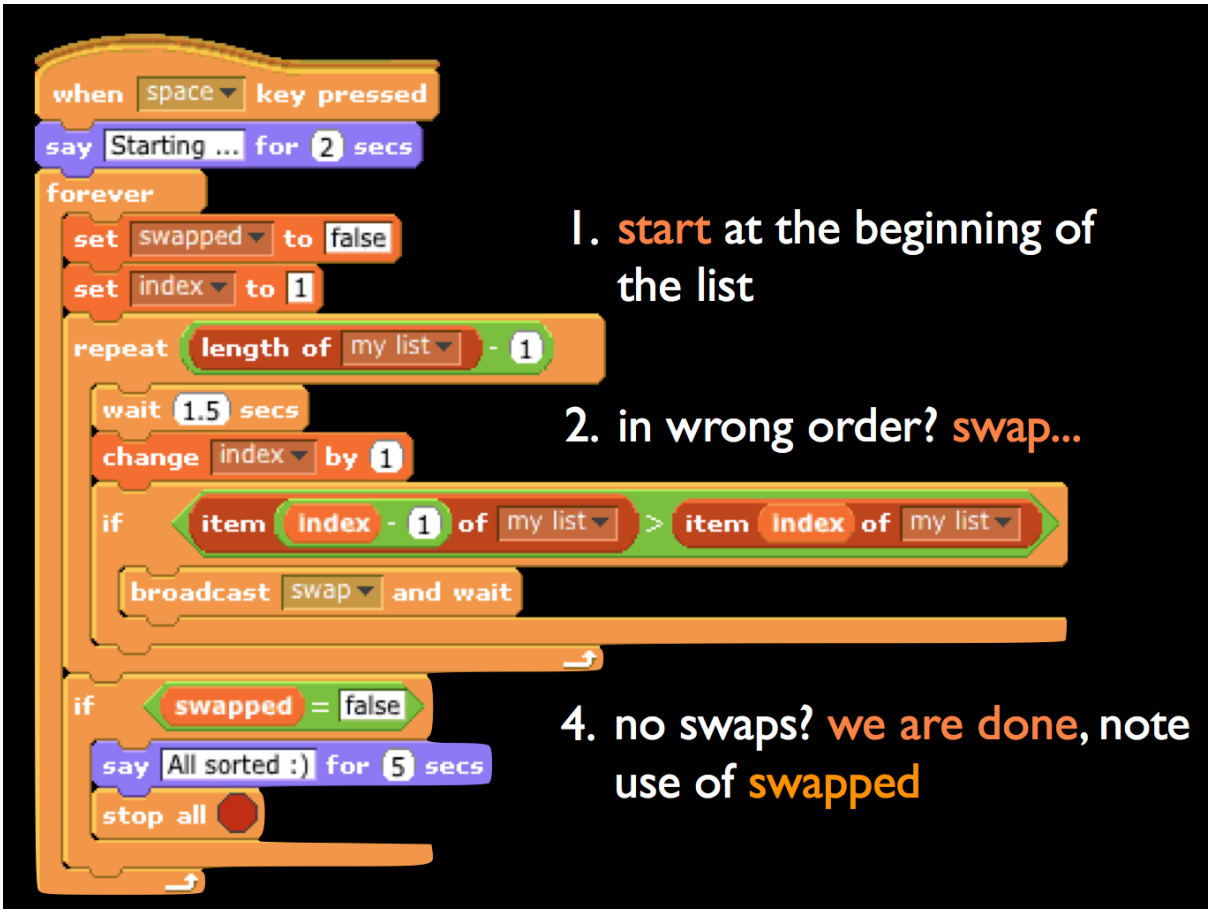
Then extend your program to do the following:

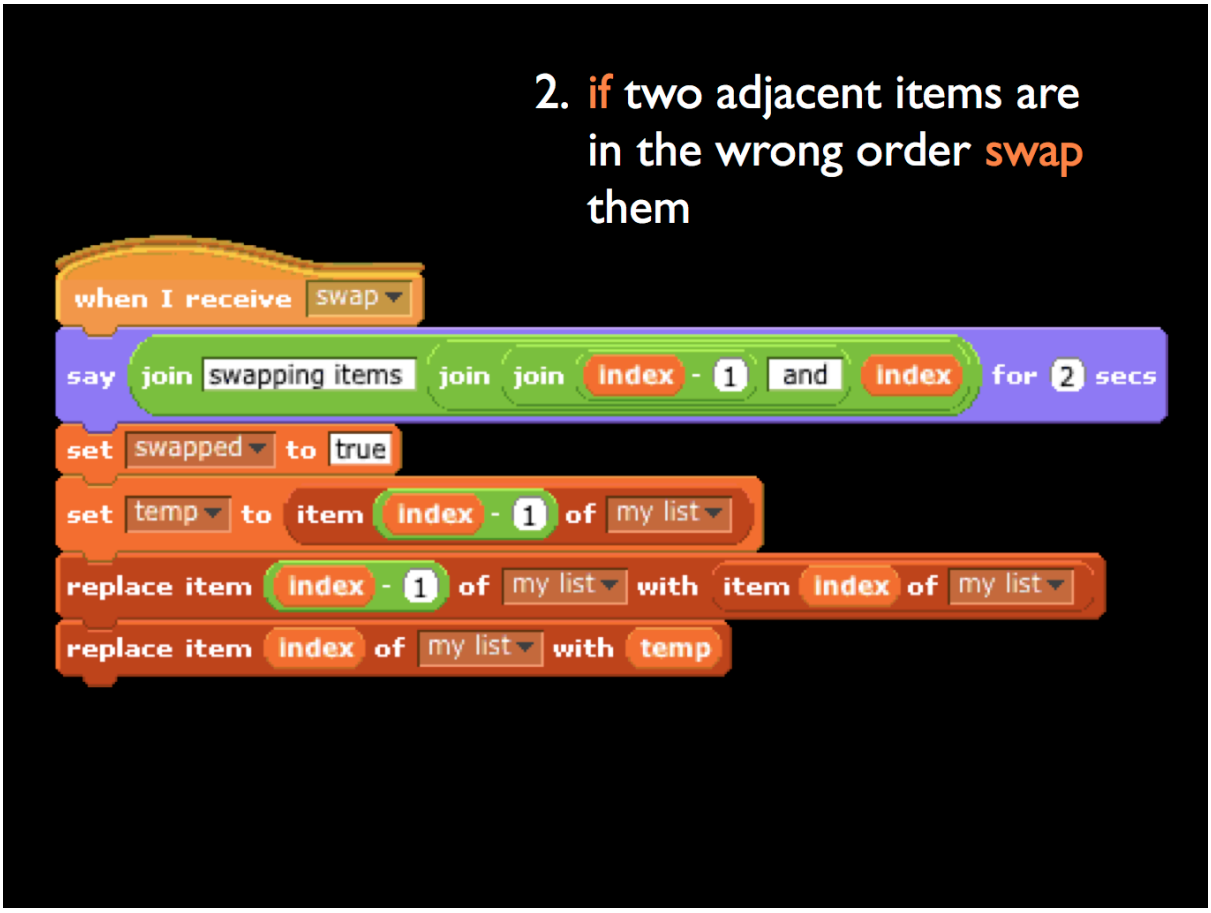
1. print out all of the numbers in the array
2. print the smallest number in the array
3. print the largest number in the array
4. print the position in the array of the smallest and largest numbers
5. calculate the arithmetic mean (*recall, the mean is the sum of the numbers di- vided by 'n' the number of numbers*)

Approach this week
Think about the overall design of each extension task. We *highly recommend sketching out each algorithm on paper first*.

Hacker edition
'Bubble sort' is a well-known sort algorithm that is used to put a list of items into ascending order. An example implementation in Scratch is shown overleaf, and there are various explanations online (pseudocode).

1. Write a C implementation of bubble sort.
2. Count the number of times through the loop and the number of swaps made, and output this at the end of the program.
3. (optional) Generate random lists to sort, sort and output them, examine the performance of the algorithm for 10 different random lists.


when space key pressed
say Starting ... for 2 secs
forever
set swapped to false
set index to 1
repeat length of my list - 1
wait 1.5 secs
change index by 1
if item index - 1 of my list > item index of my list
broadcast swap and wait
if swapped = false
say All sorted :) for 5 secs
stop all
1. start at the beginning of the list
2. in wrong order? swap...
4. no swaps? we are done, note use of swapped


2. if two adjacent items are in the wrong order swap them
when I receive swap
say join swapping items join join index - 1 and index for 2 secs
set swapped to true
set temp to item index - 1 of my list
replace item index - 1 of my list with item index of my list
replace item index of my list with temp