Lab 6: A Fourth C Program

Aim

This lab class will provide you with an opportunity to develop your C programming skills and in particular:

- Construct a program consisting of multiple functions (some returning values, some 'altering' parameters);
- Experimenting with `if` statements, `while` and `for` loops; and
- Utilising input (`scanf()`), output (`printf()`), and arrays.

Context

The user has a bunch of numbers in triples. For each triple, they want to know what the middle value (in magnitude) is. For example, if the user has the numbers 15, 3, and 20, then the middle value is 15.

Task

1. Create a new C project with name `Lab6` as explained on the link accompanying the *Tutorial 4* question on MyLO. (Although you may use any C editor, it is recommended that you learn how to use **Microsoft Visual Studio** as this is the system used for assignments).

2. Within the project create a C source file called `middling.c`

3. Add code to the `main()` function which asks the user to enter a `char` value, obtains a `char` value and repeatedly does this as long as the `char` is `'y'`.

4. Add a function called `get3ints()` which has the following function header:
   ```
   void get3ints(int *a, int *b, int *c)
   ```
   The function should ask the user for — and obtain — three `int` values. Each is entered by pressing the <*Enter*> key, but this will confuse the code from step 3. To resolve this, when the third number is obtained a `char` should be obtained too. The appropriate format string is `"%d%c"`.

5. Add code to the `main()` function to test `get3ints()`.

6. Add a function called `middle()` which has the following function header:
   ```
   int middle(int a, int b, int c)
   ```
   It should determine the middle numerical value of the three given values. If two or more values are identical it should still work. For example with numbers 3, 9, and 5 the middle value is 5; with numbers 2, 2, and 1 the middle value is 2; *etc*.

   You should implement the `middle()` function using the following algorithm:
   - Create an array consisting of the values given on the parameter list
   - Sort the array by:
     - Using a loop from 0–2 inclusive (with loop counter *i*)
     - Nesting a second loop from *i*–2 inclusive (with loop counter *j*)

    - Swapping array elements with indices *i* and *j* if element *j* is smaller than element *i*
- Returning the value now in the middle element of the array.

7. Add code to the `main()` function to test `middle()`.

Sample output is shown below.

Execution 1:

```
WHERE'S THE MIDDLE?

Do you want to play (y/n)? n
Thanks for playing.
```

Execution 2:

```
WHERE'S THE MIDDLE?

Do you want to play (y/n)? y
Enter int #1: 78
Enter int #2: 90
Enter int #3: 45
The middle value is : 78

Do you want to play again (y/n)? y
Enter int #1: 23
Enter int #2: 1
Enter int #3: 67
The middle value is : 23

Do you want to play again (y/n)? y
Enter int #1: 34
Enter int #2: 2
Enter int #3: 2
The middle value is : 2

Do you want to play again (y/n)? n
Thanks for playing.
```