1. **[4points] Write a C program to reverse the array.**
a) Declare an array of integers of size `10`. In your program, use the preprocessor `#define` directive to create an `SIZE` constant that has the value `10`.
b) Fill the array with pseudo-random numbers from `0` to `100`.
c) Print all values from the array to the screen.
d) Declare two variables `start` and `end`.
Use these variables and a `while` loop to reverse the values in the array as follows:
-replace the first element with the last element,
-replace the second element with the last but one element,
-etc.
With what values should start and end be initialized?
How should these variables change in the loop?
When should the loop stop?

e) Print all values from the array to the screen.

Test data:

```
Elements in array are:
arr[0] = 35
arr[1] = 57
arr[2] = 5
arr[3] = 75
arr[4] = 61
arr[5] = 99
arr[6] = 30
arr[7] = 24
arr[8] = 62
arr[9] = 86

Elements in array are:
arr[0] = 86
arr[1] = 62
arr[2] = 24
arr[3] = 30
arr[4] = 99
arr[5] = 61
arr[6] = 75
arr[7] = 5
arr[8] = 57
arr[9] = 35
```

2. **[6points] Write a C program that implements the following variation of bubble sort.**
a) Declare an array of integers of size `20`. In your program, use the preprocessor `#define` directive to create an `SIZE` constant that has the value `20`.
b) Fill the array with pseudo-random numbers from `0` to `100`.
c) Print all values from the array to the screen.
d) Declare two variables, one will count `swap`s and the other will change its value if a `swap` has occurred.

e) Use two nested `for` loops to implement bubble sort. The indices of both loops start from zero.
f) In the inner loop, we compare the adjacent array elements. If the ascending order is not satisfied, we `swap` the values in these array elements.
g) If there has been a swap, we increase the swap counter and change the value of the flag which remembers whether the elements in the array have been swapped or not.
h) We break the outer loop if there was no swap.
i) Print all values from the array to the screen.
j) Print out the number of swaps.

Test data:

```
Elements in array are:
arr[0] = 25
arr[1] = 2
arr[2] = 21
arr[3] = 23
arr[4] = 15
arr[5] = 62
arr[6] = 22
arr[7] = 23
arr[8] = 9
arr[9] = 24
arr[10] = 54
arr[11] = 98
arr[12] = 26
arr[13] = 39
arr[14] = 79
arr[15] = 50
arr[16] = 15
arr[17] = 79
arr[18] = 21
arr[19] = 2

Elements in sorted array are:
arr[0] = 2
arr[1] = 2
arr[2] = 9
arr[3] = 15
arr[4] = 15
arr[5] = 21
arr[6] = 21
arr[7] = 22
arr[8] = 23
arr[9] = 23
arr[10] = 24
arr[11] = 25
arr[12] = 26
arr[13] = 39
arr[14] = 50
arr[15] = 54
arr[16] = 62
arr[17] = 79
```

```
arr[18] = 79
arr[19] = 98

Number of swaps 77
```

3. **[2points] Write a C program that implements the following variation of bubble sort.**
a)Replace the outer `for` loop with a `do-while` loop.

Test data:
See exercise 2.

4. **[2points] Write a function to which we pass 4 values, and it returns the sum of the minimum and maximum.**

a) Define a function which we pass 4 integer values and which returns an integer.
b) We only have 5 independent `if`s in the function body.
c) If the condition in `if` is true, we do the appropriate swap of the values of the variables.
d) Do not use `else`.
e) Inside the function print the values of the sorted variables. Inside the function, you can use the `printf` function only once.
f) Return the sum of the minimum and maximum.
g) In the main function, test the written function for the following cases:
`1,2,3,4`
`14,12,13,11`
`24,23,22,21`
`h)` Do not use any variables inside the main function.

Test data:
```
a=1, b=2, c=3, d=4
sum of min+max=5

a=11, b=12, c=13, d=14
sum of min+max=25

a=21, b=22, c=23, d=24
sum of min+max=45
```

5. **[4points] Write a C program that will test the functions that find the minimum and maximum.**
a)Using the ternary operator, write a function that returns the minimum value of the two variables passed to it.
b)Using the ternary operator, write a function that returns the maximum value of the two variables passed to it.
c) The body of each function is just one line.
d) Create a declaration and a definition of both functions in the program.
e) In the main function, declare two variables and read their values from the keyboard using the `scanf` function.
f) Call both functions and assign the returned values to the new variables. Print minimum and maximum to the screen.

Test data:

```
Enter any two numbers: 12 32
Maximum = 32
Minimum = 12

Enter any two numbers: 34 11
Maximum = 34
Minimum = 11
```

Extra credits [6points] : Modify the program in Exercise 2 to answer the following questions.
Which array has the minimum number of swaps in bubble sort?
Which array has the maximum number of swaps in bubble sort?
What is the average number of swaps in an array of n elements? How can this be tested experimentally?

laboratory 08 – Functions, passing arrays to functions.