CSE 102 : Structured Programming Lab
Lab Manual - 7

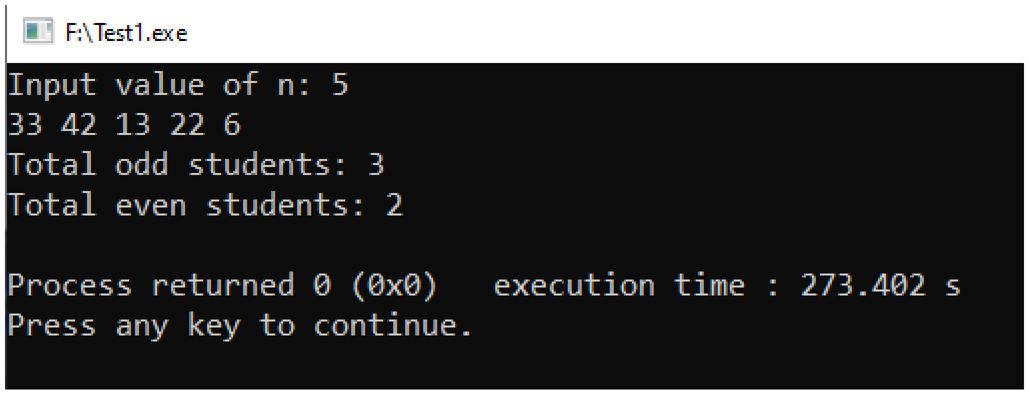
Objectives

- Problem solving using **arrays**

Task 1

John Doe is a first-grade student. He was newly admitted into X High School and started making new friends. One day in class, he was being taught about odd and even numbers. John Doe has a lot of friends, and he wants to know how many of his friends have even roll numbers and how many of his friends have odd roll numbers.

Help John Doe by writing a C program that takes the number of friends n that John Doe has as input. Then the program takes n numbers as input from the user and stores them in an array. Then the program outputs how many numbers are even and how many numbers are odd.



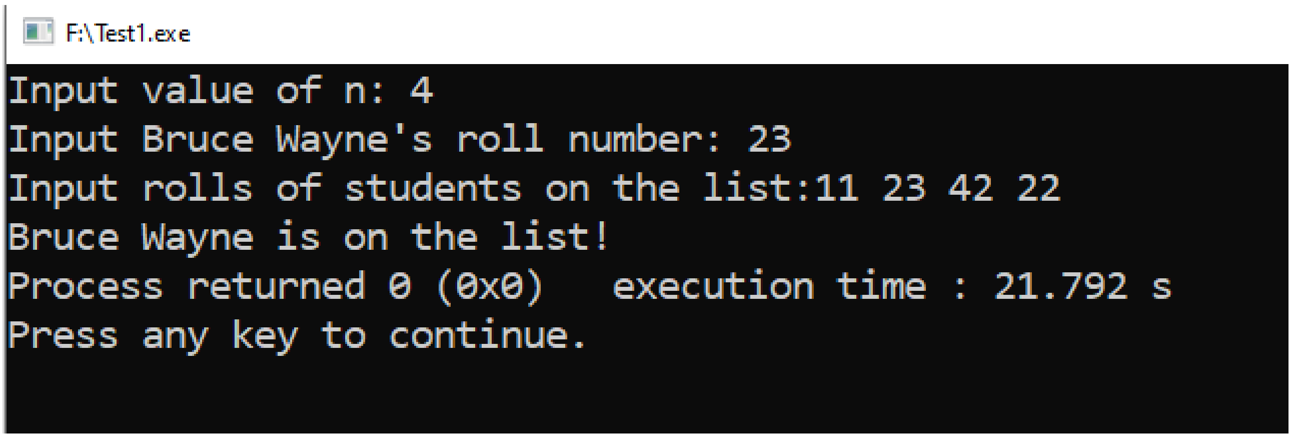
```
Input value of n: 5
33 42 13 22 6
Total odd students: 3
Total even students: 2

Process returned 0 (0x0)   execution time : 273.402 s
Press any key to continue.
```

Figure 1: Sample input and output for Task 1

Task 2

One day, John Doe's class teacher came to class with a list of roll numbers of students who performed poorly in the class test. The teacher will call the parents of the students on that list. John Doe wants to know if his best friend Bruce Wayne is on that list.

Help John Doe by writing a C program that takes the number of students on that list as n. Then the program takes the roll number of John Doe's friend Bruce Wayne as input. Then the program will search whether Bruce Wayne's roll number is on that list. If Bruce is on the list, the program will output: `Bruce Wayne is on the list!` and otherwise, the program will output: `Bruce Wayne is not on the list!`.



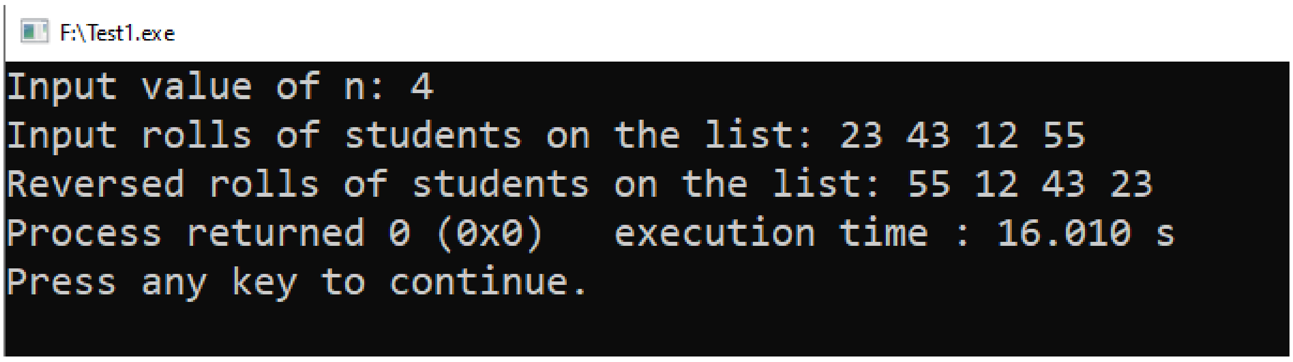
```
F:\Test1.exe
Input value of n: 4
Input Bruce Wayne's roll number: 23
Input rolls of students on the list:11 23 42 22
Bruce Wayne is on the list!
Process returned 0 (0x0)   execution time : 21.792 s
Press any key to continue.
```

Figure 2: Sample input and output for Task 2

Task 3

John Doe is going on a school field trip. His teacher gave a list of student roll numbers that correspond to their bus seats. John Doe has a naughty plan in mind and wants to reverse the order of the seats from the list that his teacher gave.

Help John Doe by writing a C program that takes the number of students on that list as n. Then the program takes the roll number of the students who are going on the field trip as input. Then the program will print the exact list but in reverse order.



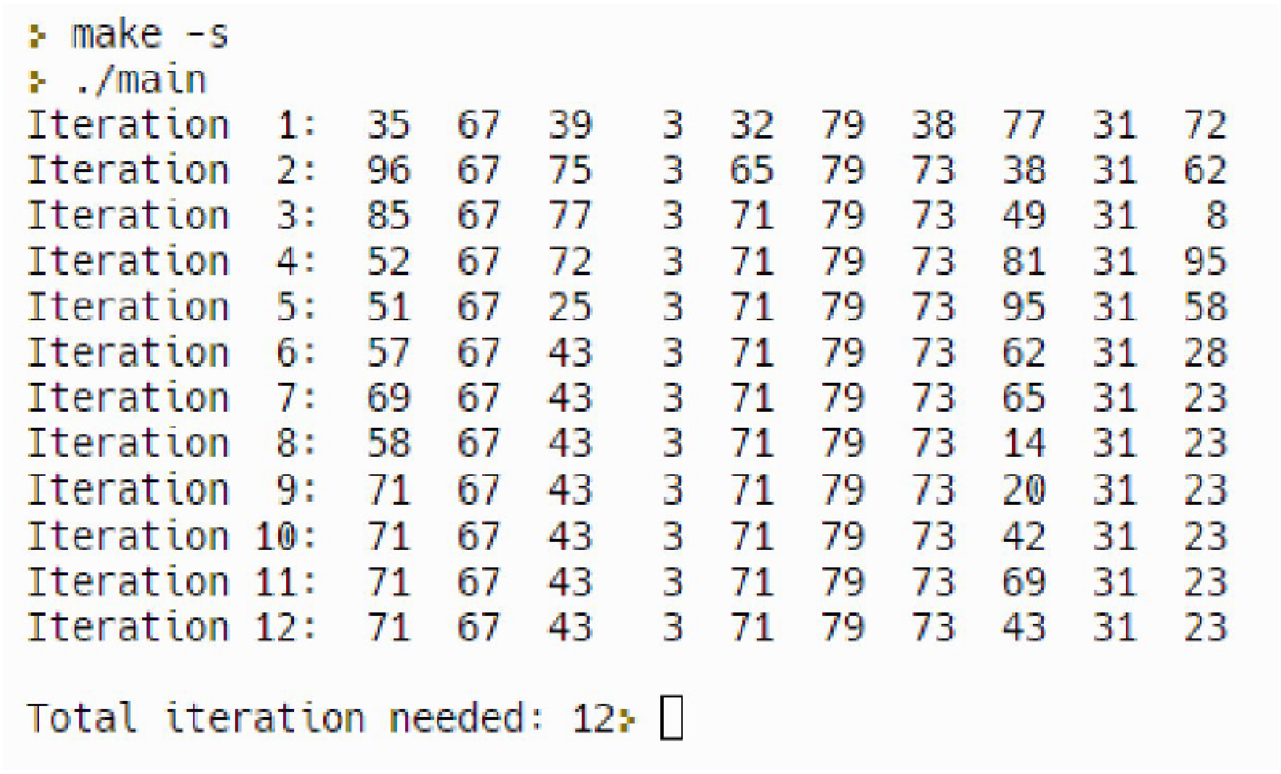
```
F:\Test1.exe
Input value of n: 4
Input rolls of students on the list: 23 43 12 55
Reversed rolls of students on the list: 55 12 43 23
Process returned 0 (0x0)   execution time : 16.010 s
Press any key to continue.
```

Figure 3: Sample input and output for Task 3

Task 4

Write a C program that generates 10 random numbers ranging from 1 to 100. The numbers should be stored in a one-dimensional array of size 10. You have to keep traversing (visiting the array) the array and replace the non-prime numbers with another random value. You need to keep on performing this operation until all the numbers become prime numbers. You have to print the contents of the array in each iteration and finally print how many iterations were needed to complete the task.

N.B. You guys are free to take help from the internet for this problem.



```
> make -s
> ./main
Iteration  1:  35  67  39   3  32  79  38  77  31  72
Iteration  2:  96  67  75   3  65  79  73  38  31  62
Iteration  3:  85  67  77   3  71  79  73  49  31   8
Iteration  4:  52  67  72   3  71  79  73  81  31  95
Iteration  5:  51  67  25   3  71  79  73  95  31  58
Iteration  6:  57  67  43   3  71  79  73  62  31  28
Iteration  7:  69  67  43   3  71  79  73  65  31  23
Iteration  8:  58  67  43   3  71  79  73  14  31  23
Iteration  9:  71  67  43   3  71  79  73  20  31  23
Iteration 10:  71  67  43   3  71  79  73  42  31  23
Iteration 11:  71  67  43   3  71  79  73  69  31  23
Iteration 12:  71  67  43   3  71  79  73  43  31  23

Total iteration needed: 12>
```

Figure 4: Sample input and output for Task 4

Hint: Use the `rand()` function to generate random numbers. You have to use the `stdlib.h` library function to use the `rand()` function. The `rand()` function can generate very large numbers that will not be sufficient for this task. To get a random number between 1 and 100, divide the number by 100 and use the modulus operator to get the remainder. Since you are dividing the number by 100 and getting the remainder, the remainder will never be greater than 100.

Sample code showing the usage of the `rand()` function is given below.

```
#include<stdio.h>
#include<stdlib.h>

int main(){
    int random_num1 = rand();
    int random_num2 = rand() % 100 + 1;

    printf("Random number: %d\nRandom number between 1 and
    100: %d", random_num1, random_num2);

    return 0;
}
```