EECS2031 Assignment 1

Due: Oct 25 (Monday) 11:59 pm Total marks: 100 pts

In this assignment (also called Program Exam in some courses), you are going to write a few ANSI C programs. Note that unlike the weekly labs, this is an individual work, and you should not discuss with others. Ask the instructor for help.

Question 1

Question 1A getchar, int literals, arrays (20 pts)

Specification

Complete an ANSI-C program, which should use `getchar` (**only**) to read and interpret integer.

Implementation

Download the partially implemented file `a1A.c` and start from there.
The program:

- should use `getchar()` to read inputs (from standard in) until EOF is read. The input contains decimal integer literals separated by one blanks or new line characters. The program interprets each literal and put into an int array. When "end of file" is reached, the program prints out the value of each integer and the double of the value.
- assume all literals in the input are valid decimal `int` literals. Note that the input can contain negative numbers.
- should not use other input IO functions such as scanf(), fgets(). Only `getChar`() is allowed to read input. (So have to read char by char.)
- should not use other library function such as atoi() (So have to convert manually)
- should not use extra array. Only use one array resu[].
- as mentioned in lab, p43 of the K&R book contains an example of interpreting int literals char by char (on the fly).

Sample Inputs/Outputs: (download – don't copy/paste - the input file `inputA.txt`)

```
red 306 % gcc a1A.c -o a1A
red 306 % a1A
3 12 435
54 -15
7 98 -10 456
^D
--------
3       6
12      24
435     870
54      108
-15     -30
7       14
98      196
```

```
-10      -20
456      912
red 308 % a1A < inputA.txt
--------
5        10
34       68
534      1068
-12      -24
43       86
-13      -26
7        14
54       108
-122     -244
3245     6490
red 309 %
```

Submit your program by issuing **`submit 2031AC a1 a1A.c`**
Or via websubmit (see end of this pdf for more info)

Question 1B getchar, floating point literals, arrays (40 pts)

Specification

Extend the program in 1A, so that it uses `getchar` (**only**) to read and interpret floating point literals.

Implementation

Name your program `a1B.c`. The program:

- should use `getchar()` to read inputs (from standard in) until EOF is read. The input contains floating point literals and integer literals separated by one blanks or new line characters. The program interprets each literal and put into a float array. When "end of file" is reached, the program prints out the value of each float as well as the value multiplied by 2, as shown in sample output below.
- assume all floating point literals are valid `float/double` or `int` literals (both `2.3, 5, .4` are considered valid). There are no negative numbers.
- should not use other input IO functions such as scanf(), fgets(). Only `getChar`() is allowed to read input. (So have to read char by char.)
- should not use other library function such as atoi(), atol(), atof(), strtol(), strtoul(). (So have to convert manually)
- should not use extra array. Only use one array resu[].

Sample Inputs/Outputs: (download – don't copy/paste - the input file `input2E.txt`)

```
red 306 % gcc a1B.c -o a1B
red 306 % a1B
2.3 4.56
43.3 43 5.3
.3 1.2
^D
--------
```

```
2.3000  4.6000
4.5600  9.1200
43.3000 86.6000
43.0000 86.0000
5.3000  10.6000
0.3000  0.6000
1.2000  2.4000
red 307 % cat inputB.txt
3.25 24.54
4.323 4.54
.4
1 0.29
red 308 % a1B < inputB.txt
--------
3.2500  6.5000
24.5400 49.0800
4.3230  8.6460
4.5400  9.0800
0.4000  0.8000
1.0000  2.0000
0.2900  0.5800
```

Submit your program by issuing **`submit 2031AC a1 a1B.c`**
Or via websubmit (see end of this pdf for more info)

Question 2 Maintaining sorted array (40pts)

Specification

Operate on an array of integers that is initially sorted in non-decreasing order. The array must always be maintained in non-decreasing order. The array's capacity is indicated by MAX_SIZE and its current size is indicated by 'size'.

Implementation

Given an array *arr* of integers that is sorted, and is maintained by its 'size', implement the following functions:

- `myAdd` (`arr,..,d`): add an integer *d* to the array; return a number ≥ 0 if the operation is successful; return an negative number if the operation is unsuccessful. The adding is unsuccessful if, before insertion, the MAX_SIZE has reached.
- `myRemove` (`arr,..,d`): remove integer *d* from the array; return a number ≥ 0 if the operation is successful; return a negative number otherwise (e.g., *d* is not found in the array). Assume no duplicate values in the array.
    - Note that you don't need to remove *d* from the memory. All you need is to make sure that d does not show in the output of the current 'size' elements. And all existing element maintained sorted.
- `myBinarySearch` (`arr,..,d`): given an integer *d*, if *d* is found in the array, return the index of the cell containing *d*. Return a negative number otherwise (e.g., *d* is not found in the array). Assume no duplicate values in the array.

- Note that since the array is maintained sorted, instead of using linear search which has complexity `O(n)`, you need to do binary search, which has complexity `log(n)`. You should not use C's library function to do the search. Write you own binary search.

Sample Inputs/Outputs: **Here we assume the MAXSIZE is 5. In your submitted code, the MAXSIZE should be the given value 20.**
Also you may want to test more cases.

```
red 308 ./a.out
a 10
[ 10 ]
a -98
[ -98 10 ]
a 5
[ -98 5 10 ]
a 67
[ -98 5 10 67 ]
s 5
Found 5 at index 1.
s 10
Found 10 at index 2.
a -67
[ -98 -67 5 10 67 ]
a 90
Failed to add 90.
r 67
[ -98 -67 5 10 ]
r -1
Failed to remove -1.
s -5
Not found -5.
r -67
[ -98 5 10 ]
r 5
[ -98 10 ]
r 5
Failed to remove 5
r -98
[ 10 ]
r 10
[ ]
r 10
Failed to remove 10.
s 6
Not found 6.
a 17
[ 17 ]
q 100
red 309
```

Submit your program using **`submit 2031AC a1 arraySorted.c`**

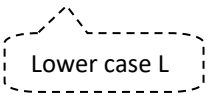
Or via websubmit (see end of this pdf for more info)

End of assignment

Make sure your program compiles in the lab environment. The program that does not compile, or generate 'segmentation fault' in the lab, will get 0.

All submissions need to be done from the lab. No late submission will be accepted. No makeup assignment.

In summary, for this assignment you should submit:
a1A.c a1B.c arraySorted.c
At any time and from any directory, you can issue **submit -l 2031AC a1** to view the list of files that you have submitted.



Common Notes

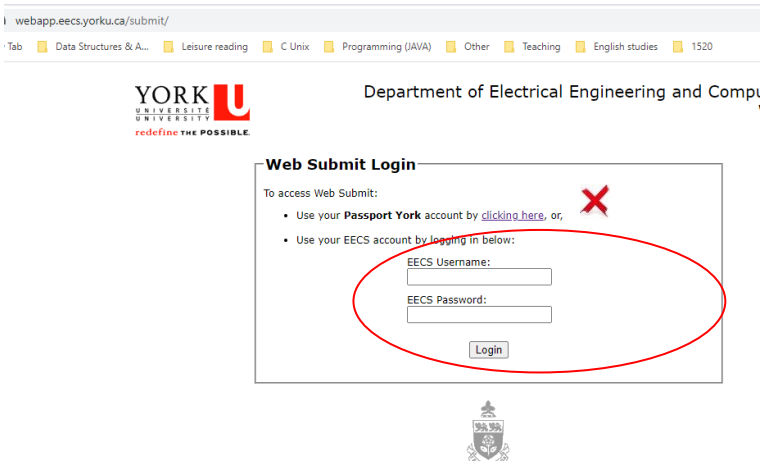
All submitted files should contain the following header:

```
/**************************************
* EECS2031AC – Assignment 1 *
* Author: Last name, first name *
* Email: Your email address *
* EECS_username: Your EECS login username *
* York num: Your YorkU student number
**************************************/
```

Web-submit

To prepare for the coming labtest, for which you will be encouraged to work on your local computer and submit using websumit, for this assignment you can submit using websubmit. So, in addition to submitting from the lab, you can also submit from local machine directly. (You should still make sure that the files compile in the lab environment)
You can submit using websubmit at **https://webapp.eecs.yorku.ca/submit/**

- **login using EECS , not passport York. If you keep on getting prompted to login using passport York, then clear the cookie and cache of your browser and then try again. Eventually you should see the page below.**
- **Once login, select Course: 2031AC and then select Assignment: a1**
- **Then upload files from your local computer.**



Use EECS account

See this page for more information about web-submit
https://wiki.eecs.yorku.ca/dept/tdb/services:submit:websubmit