COMP281 2017 – Assignment 3

- In the following, you will find the problems that constitute Assignment 3. They will be also available on the *online judging (OJ)* system available at http://intranet.csc.liv.ac.uk/JudgeOnline/

- You need to write a valid C program that solves each of these problems – it must read the input, as specified in the problem description then print the solution to the given problem for that input.
  - Note that code is "correct" only if it **correctly implements a solution to the problem stated** in the assignment, not "if online judge accepts it".
  - That is, even if OJ accepts your code, it could be wrong. Read the problems carefully.

- Input is read from the standard input, in the same way that you read input from the keyboard as shown in lectures (e.g., using scanf). Output is also printed to the standard output, as you have seen (e.g., using printf).

- When you are satisfied that your programs work correctly, you must submit them through the departmental submission system, as described below.
  - Even if the program is not correct, still submit whatever you have! You can still earn points if certain parts of the code are done right.

- You must also include a brief report describing your solutions to the problems. This should be maximum two sides of A4 paper, 10pt font. Shorter is preferred over longer, but do mention all important design choices you made for you solutions. This should include describing the algorithm used to reach the solution, describing your use of any C language features (that were not discussed in lectures) and identifying any resources that you have used to help you solve the problems.

  There will be a penalty of 5% of the total points deducted for over-length reports, and any pages past page 4 will not be read.

- This assignment is worth 40% of the total mark for COMP281.
  - All problems are weighted equally.
  - For each problem, you can earn a total of 20 points
    - 10 points for "Functionality and Correctness" awarded for programs that **correctly** solve the problem for all test cases.
    - 8 points for "Programming style, use of comments, indentation and identifiers" awarded depending on the style, comments and efficiency of the solution
    - 2 points for the quality and depth of the accompanying report
  - The final grade results from normalizing the earned points to a scale of 100.
  - See separate "COMP 281 Detailed Marking Guidelines" for more details.

Submission Instructions

- Name each of your c files according to the problem number; i.e., problem 10xx should be in a file 10xx.c (where 'xx' is replaced by the actual number).
- Place all of your C files, one per problem, and your report (in .pdf format) into a single zip file.
    - Please use the standard (pkzip) zip file format:
      https://en.wikipedia.org/wiki/Zip_%28file_format%29
      which is supported by winzip, winrar, etc. on windows/mac os X, and 'zip' on linux
    - test your zip file before submitting.
- Before you submit your solutions, please first test them using the online judge. **You are required to include the "Result" and the "RunID" in your C code as comments.** The OJ provides a RunID. **RunIDs are not problem IDs**.
    - Example:
        - the problem is 10xx
        - The solution has been accepted by the OJ, and the runID is 2033.
        - You add to your code: /* 2033 Accepted */ to 10xx.c
    - Result is one of the following: Accepted, Wrong Answer, Presentation Error, Time Limit Exceeded, Memory Limit Exceeded, Output Limit Exceeded, Runtime Error, Compile Error
- Submit this zip file using the departmental submission system at
  http://www.csc.liv.ac.uk/cgi-bin/submit.pl
  Only the file submitted through this link will be marked.
- The **deadline** for this assignment is **Wed March 15, 2017, 4:30pm**
- Penalties for late submission apply in accordance with departmental policy as set out in the student handbook, which can be found at:
  http://intranet.csc.liv.ac.uk/student/ug-handbook.pdf

1. Problem 1048

Title: Sort and Search for Dates II

Description

Input n (1<=n<=10000) and then followed by n lines. Each line corresponds to a **valid** date, consisting of one string ("January", "February", ..., or "December"), one integer between 1 and 31, and one two digit integer representing the year (from 90 to 99, and then from 00 to 12). You do not have to worry about date validation. All dates in the input are valid dates.

Please use structures to store the dates. Please use malloc to dynamically allocate just enough space for n structures. You are asked to sort the dates chronologically using the built-in qsort function. Please output the sorted list, one date per line, from most recent date to oldest. Please also use the built-in bsearch function to allow for a user query to check whether a specific date is in the list, and output either "Yes" or "No".

Input

- n, the number of dates to sort,
- followed by n dates,
- followed by a user query in format *day month year* (e.g. “1 1 00” or “31 3 68”).
  (Note this is a different format as the rest of the dates are presented in)

Output

- Sorted list of dates,
- followed by “Yes” or “No” to indicate whether the query date input by the user (e.g. 1 1 00 *day month year*) is in the list.

Sample Input

```
10
January 1 01
January 1 00
February 28 99
July 17 12
September 10 12
July 1 00
June 30 90
August 25 06
May 27 08
October 1 03
1 1 00
```

Sample Output

```
September 10 12
July 17 12
May 27 08
August 25 06
October 1 03
January 1 01
July 1 00
January 1 00
February 28 99
June 30 90
```
Yes

2. Problem 1082

Title: Sort either the odd or the even numbers

Description

You will need to process a (potentially large) list of integers. The goal is to either sort (into ascending order) all the even number in the list, or all the odd numbers. Which numbers you need to sort is indicated by the first number of the input. The numbers of the non-indicated parity should be unaltered.

E.g, if you are instructed to sort the odd numbers, then print the list with all even numbers (including 0) in their original place in the input and the odd numbers sorted into ascending order.

Input

The input consists of 3 lines:

-the first line contains the word 'odd' or 'even' and indicates which numbers you need to sort

-the second line contains the length of the list you will need to process

-the 3rd line contains the list of integers.

Output

The processed input with either odd or even numbers sorted.

Sample Input

```
odd
10
5 3 4 2 1 6 8 7 10 9
```

Sample Output

```
1 3 4 2 5 6 8 7 10 9
```

3. Problem 1084

Title: Highway Lite

=Overview=

In this problem you are asked to implement a simplistic highway simulation. The 'highway' is (rows x cols) grid of cells that can be occupied by a vehicle or not. All traffic starts at the left side of of the grid (at column 0) and travels right on, never switching lane (i.e., row). If a car moves beyond the last column it exits the simulation. As input you will be given a list of arrival times of vehicles, the goal is to produce the state of the highway after a specified number of simulation steps.

=The Highway=

The highway is a (rows x cols) grid, where row 0 corresponds to the topmost lane, and column 0 is the leftmost segment. I.e., a 3 lane highway of 10 segments looks like this:

```
..........    <- row 0, is the topmost lane
..........
..........

^
|
column 0, is the leftmost segment
```

=Vehicles=

In this version, there is just 1 vehicle type: all vehicles get to move 1 step to the right in each time step.

=Movement, Priorities & Arrivals=

The simulation is executed in discrete time steps. Each step consists of 2 phases:

1) the movement phase, and
2) the arrival phase.

In the movement phase, all the vehicles that are already on the highway get to move. They do not move synchronous, but one after another: vehicles that entered the simulation first get to move first.

After all vehicles already on the board have been given the opportunity to move, new vehicles can arrive. These are read from a list that serves as the program input.

=Detailed 1 lane Example=

Here we provide a detailed example of how the simulation should be executed. Suppose this is the input:

```
1 50 40
2 0
5 0
9 0
11 0
```

```
14 0
20 0
21 0
22 0
23 0
24 0
25 0
26 0
28 0
30 0
32 0
34 0
36 0
38 0
```

Then the following simulation should occur:

(Note, the line after the indicated time step shows the state at the beginning of that time step. So, the first car arrives in the arrival phase of timestep t=2 and therefore is shown at the beginning of time step t=3)

```
t=0
................................................
t=1
................................................
t=2
................................................
t=3
1...............................................
t=4
.1..............................................
t=5
..1.............................................
t=6
1..1............................................
t=7
.1..1...........................................
t=8
..1..1..........................................
t=9
...1..1.........................................
t=10
1...1..1........................................
t=11
.1...1..1.......................................
t=12
1.1...1..1......................................
t=13
.1.1...1..1.....................................
t=14
..1.1...1..1....................................
t=15
1..1.1...1..1...................................
t=16
.1..1.1...1..1..................................
t=17
```

```
..1..1.1...1..1................................
t=18
...1..1.1...1..1...............................
t=19
....1..1.1...1..1..............................
t=20
.....1..1.1...1..1.............................
t=21
1.....1..1.1...1..1............................
t=22
11.....1..1.1...1..1...........................
t=23
111.....1..1.1...1..1..........................
t=24
1111.....1..1.1...1..1.........................
t=25
11111.....1..1.1...1..1........................
t=26
111111.....1..1.1...1..1.......................
t=27
1111111.....1..1.1...1..1......................
t=28
.1111111.....1..1.1...1..1.....................
t=29
1.1111111.....1..1.1...1..1....................
t=30
.1.1111111.....1..1.1...1..1...................
t=31
1.1.1111111.....1..1.1...1..1..................
t=32
.1.1.1111111.....1..1.1...1..1.................
t=33
1.1.1.1111111.....1..1.1...1..1................
t=34
.1.1.1.1111111.....1..1.1...1..1...............
t=35
1.1.1.1.1111111.....1..1.1...1..1..............
t=36
.1.1.1.1.1111111.....1..1.1...1..1.............
t=37
1.1.1.1.1.1111111.....1..1.1...1..1............
t=38
.1.1.1.1.1.1111111.....1..1.1...1..1...........
t=39
1.1.1.1.1.1.1111111.....1..1.1...1..1..........
t=40
.1.1.1.1.1.1.1111111.....1..1.1...1..1.........
```

As such, the requested output in this case would be:

```
.1.1.1.1.1.1.1111111.....1..1.1...1..1.........
```

Input

First a row with 3 integers:

```
number_of_rows number_of_colums number_of_timesteps
```

that specify the size of the highway and the number of steps to simulate.

This is followed by the aforementioned list of arrivals. Each entry is a row with 2 integers:

```
arrival_time row_index
```

-These rows will be ordered based on arrival time.

Output

The state of the highway after number_of_timesteps have been simulated, encoded as follows:

'.' denotes empty space
'1' denotes a vehicle

Sample Input

```
5 10 20
0 1
0 3
0 4
2 4
4 3
4 4
6 3
6 4
10 0
10 1
12 2
12 4
18 1
```

Sample Output

```
.........1
.1.......1
.......1..
..........
.......1..
```

4. Problem 1083

Title: Highway

=Overview=

In this problem you are asked to implement a simplistic highway simulation. The 'highway' is (rows x cols) grid of cells that can occupy a vehicle or not. All traffic starts at the left side of of the grid (at column 0) and travels right on, never switching lane (i.e., row). If a car moves beyond the last column it exits the

simulation. As input you will be given a list of arrival times of vehicles, the goal is to produce the state of the highway after a specified number of simulation steps.

=The Highway=

The highway is a (rows x cols) grid, where row 0 corresponds to the topmost lane, and column 0 is the leftmost segment. I.e., a 3 lane highway of 10 segments looks like this:

```
..........    <- row 0, is the topmost lane
..........
..........

^
|
column 0, is the leftmost segment
```

=Vehicles=

To make the problem a bit more interesting, there are 3 types of vehicles: cars, trucks, and campers. These can travel 3, 1 and 2 cells per time step, maximally; if there is another vehicle ahead of the vehicle it can only move to the square directly behind the leading vehicle. In other words, vehicles will not overtake each other, nor pass through another.

=Movement, Priorities & Arrivals=

The simulation is executed in discrete time steps. Each step consists of 2 phases:

1) the movement phase, and
2) the arrival phase.

In the movement phase, all the vehicles that are already on the highway get to move. They do not move synchronous, but one after another based on the vehicles priority: an integer number between 0 and 40 (40 being the highest priority) that is determined when the vehicle enters the simulation. In case of equal priorities, ties are broken based on column (lower column index is higher priority) and then based on row (lower row index is higher priority).

After all vehicles already on the board have been given the opportunity to move, new vehicles can arrive. These are read from a list that serves as the program input. Each entry of this list specifies 4 numbers:

arrival_time row_index vehicle_type priority

If a vehicle can't enter its assigned lane at the assigned arrival time (i.e., cell [row_index, 0] is occupied in the arrival phase of t=arrival_time), then the arriving vehicle is dropped from the simulation completely.

=Detailed 1 lane Example=

Here we provide a detailed example of how the simulation should be executed. Suppose this is the input:

```
1 50 100
2 0 1 26
```

```
5 0 3 26
9 0 3 26
11 0 1 34
14 0 1 17
30 0 2 26
45 0 3 27
50 0 1 28
60 0 2 28
75 0 3 27
80 0 1 26
```

Then the following simulation should occur:

(Note, the line after the indicated time step shows the state at the beginning of that time step. So, the first car arrives in the arrival phase of timestep t=2 and therefore is shown at the beginning of time step t=3)

```
t=0
..................................................
t=1
..................................................
t=2
..................................................
t=3
1.................................................
t=4
...1..............................................
t=5
......1...........................................
t=6
3.......1.........................................
t=7
..3.....1.........................................
t=8
....3.....1.......................................
t=9
......3.....1.....................................
t=10
3.......3.....1...................................
t=11
..3.....3.....1...................................
t=12
1...3.....3.....1.................................
t=13
...1..3.....3.....1...............................
t=14
.....1..3.....3.....1.............................
t=15
1......1..3.....3.....1...........................
t=16
...1.....1..3.....3.....1.........................
t=17
......1....1..3.....3.....1.......................
t=18
.........1...1..3.....3.....1....................
t=19
............1..1..3.....3.....1.
```

```
t=20
..............1.1..3......3.....................
t=21
.................11..3......3...................
t=22
...................11..3......3.................
t=23
.....................11..3......3...............
t=24
.......................11..3......3.............
t=25
.........................11..3......3...........
t=26
...........................11..3......3.........
t=27
.............................11..3......3.......
t=28
...............................11..3......3.....
t=29
.................................11..3......3...
t=30
...................................11..3......3.
t=31
2...................................11..3.......
t=32
.2.....................................11..3.....
t=33
..2.....................................11..3...
t=34
...2.....................................11..3.
t=35
....2.....................................11..
t=36
.....2.....................................1
t=37
......2....................................
t=38
.......2...................................
t=39
........2..................................
t=40
.........2.................................
t=41
..........2................................
t=42
...........2...............................
t=43
............2..............................
t=44
.............2.............................
t=45
..............2............................
t=46
3.............2............................
t=47
..3............2...........................
```

t=48
....3.....2.....
t=49
......3.....2.....
t=50
........3.....2.....
t=51
1.....3.....2.....
t=52
...1.....3.....2.....
t=53
......1.....3.....2.....
t=54
.....1.....3.....2.....
t=55
.....1.....3.....2.....
t=56
.....1....3....2.....
t=57
.....1...3...2.....
t=58
.....1..3..2.....
t=59
.....1..3.2.....
t=60
.....1.3.2.....
t=61
2.....1.3.2.....
t=62
.2.....1.3.2.....
t=63
..2.....1.3.2.....
t=64
...2.....1.3.2.....
t=65
....2.....1.3.2.....
t=66
.....2.....1.3.2.....
t=67
......2.....1.3.2.....
t=68
.......2.....1.3.2.....
t=69
........2.....1.3.2.....
t=70
.....2.....1.3.2.....
t=71
.....2.....1.3.2.....
t=72
.....2.....1.3.2.....
t=73
.....2.....1.3.2.....
t=74
.....2.....1.3.2.....
t=75
.....2.....1.3.2.....

```
t=76
3.............2.......................1.3.2....
t=77
..3............2.......................1.3.2...
t=78
....3............2.......................1.3.2..
t=79
......3...........2.......................1.3.2.
t=80
........3.........2.......................1.3.2
t=81
1........3........2.......................1.3.
t=82
...1........3.......2.......................1..
t=83
......1.......3......2.........................
t=84
.........1......3......2.......................
t=85
............1.....3.....2.....................
t=86
..............1....3....2.....................
t=87
................1...3...2.....................
t=88
..................1..3..2.....................
t=89
....................1.3.2.....................
t=90
..........................132...................
t=91
............................132.................
t=92
............................132.................
t=93
..............................132...............
t=94
..............................132...............
t=95
................................132.............
t=96
................................132.............
t=97
..................................132...........
t=98
..................................132...........
t=99
....................................132.........
t=100
....................................132.........
```

As such, the requested output in this case would be:

```
....................................132.........
```

Input

First a row with 3 integers:

```
number_of_rows number_of_colums number_of_timesteps
```

that specify the size of the highway and the number of steps to simulate.

This is followed by the aforementioned list of arrivals. Each entry is a row with 4 integers:
arrival_time row_index vehicle_type priority

-These rows will be ordered based on arrival time.
-Vehicle types are encoded as follows
1 = car
2 = truck
3 = camper

Output

The state of the highway after number_of_timesteps have been simulated, encoded as follows:

'.' denotes empty space
'1' denotes a car
'2' denotes a truck
'3' denotes a camper

Sample Input

```
5 10 20
0 1 2 15
0 3 2 14
0 4 1 34
2 4 1 7
4 3 2 18
4 4 2 24
6 3 1 11
6 4 1 8
10 0 2 40
10 1 1 23
12 2 3 11
12 4 2 34
18 1 1 3
```

Sample Output

```
.........2
...1......
..........
..........
.......2..
```

HINT

Input sizes, and highway sizes could be large, so think about how to efficiently deal with the priorities.