CENG140
Section I

C Programming

Spring 2017-2018

Take Home Exam 1

1 Regulations

- Due date: 18 April 2018, Wednesday, 23:55 Not subject to postpone, please do not ask for.
- Submission: Electronically. You will be submitting a compressed file containing your source codes through COW (More detailed description in Grading and Specifications part). Resubmission is allowed (till the last moment of the due date), the last will replace the previous.
- Cheating: We have zero tolerance policy for cheating. People involved in cheating will be punished according to the university regulations.
- Team: There is no teaming up. The take home exam has to be done/turned in individually.
- Newsgroup: You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.
- Evaluation: Your program will be evaluated automatically using black-box technique so make sure to obey the specifications. Any program that cannot collect 30 points will be graded by eye (glass-box evaluation). These eye evaluation are not open to question. They are final and decisive and are not subject to any objection or questioning.

2 Introduction

In this take home exam, you are expected to find a path for a rabbit in a carrot field. The field consists of NxN cells (maximum value of N is 100), and rabbit gains energy whenever it eats a carrot. The gain of energy is dependent to some rules, which are given below:

1. If a carrot is level l, it weights $100 \times l$ grams.
2. If a carrot weights γ grams, the gain of energy is $0.4 \times \gamma$ calories.
3. N will be given to you as input at the beginning (see sample I/O).
4. The rabbit can only move rightward and downward and each move is restricted with one cell.
5. The trip starts from the top-left corner ([0,0]) and ends at the bottom-right corner([N-1,N-1]).
6. The goal is to maximize the energy gained when the rabbit reaches the bottom-right corner.

3 Part 1 - Construction of the Field (20+40 points)

At the beginning of the program, the field is empty. Levels of the carrots that vegetate in each cell will be given to you ordered in time. You are expected to fill the field obeying some rules:

1. There are different types of carrots that are indicated by their level id. In the next section, you are going to calculate the weights of these carrots based on their level.
2. When a carrot vegetate in a cell (namely, current cell) with level x, if a link is formed among **three** or more carrots that have same level id, all of them will be gathered together the current cell, forming a new carrot with level x+1. Carrots that forming the link will be disappear and those cells will have the value 0.
3. The link mentioned above can only be formed among neighboring carrots. Neighborhood of a cell consists of cells at north,east,south and west of it (you don't need to consider diagonal relationships).
4. Forming a new carrot with a higher level may trigger another link. You should handle such cases as well.
5. Empty cells have value 0 and all cells are empty at the beginning. You should not consider the links between these kind of cells (do not create 1-level carrots using these).

3.1 Example: State of the Field by Time

There is an example of the construction of a 5x5 field in time below. Note that empty cells has value zero, which is not shown in the figures.

step 1:
empty field

step 2:
1 0 1

1				

step 3:
2 2 2

1				
		2		

step 4:
2 2 3

1				
		2		
		2		

step 5:
1 0 2

1				
1		2		
		2		

step 6:
1 1 2

	3			

step 7:
1 3 0

			1	
	3			

step 8:
1 3 1

			1	
			1	
	3			

step 9:
1 3 3

			1	
			1	
	3			
			1	

step 10:
1 4 3

			1	
			1	
	3			
			1	1

step 11:
1 3 2

	3		2	

step 12:
2 4 2

	3		2	2

step 13:
3 1 3

	3		2	2
	3			

step 14:
2 2 2

		4		

step 15:
4 2 3

		4		
		4		

step 16:
4 3 4

		4		
		4		
			4	

Here are the explanations for the figure above:

– step 1: all values are 0.

– step 2: 1 0 1 means put a carrot with level 1 into the cell with 0th column, 1st row. Nothing happens.

– step 3: 2 2 2 means put a carrot with level 2 into the cell with 2nd column, 2nd row. Nothing happens.

– ...

– step 6: 1 1 2 means put a carrot with level 1 into the cell with 1st column, 2nd row. When we do that, there a link emerges between **three** carrots with level 1 and they will form a carrot level 2 (1+1=2) at the cell [1,2]. Note that the input cell for this step is [1,2]. Since having a level 2 carrot at this cell also means that forming a line consisting of **three** level 2 carrots, they will form a level 3 carrot at this cell. Note that the intermediate steps are not shown and the carrots that form the links are disappeared except the higher level one (cells that contain disappeared carrots have value 0 now).

– ...

– step 10: Although we have four level 1 carrots, after placing the current one, they are not linked. The longest link among these have length 2. Therefore nothing happens.

– step 11: When we put a level 1 carrot into the cell [3,2] as indicated in the input, a link with length five emerges between level 1 carrots. Since 5>=**3**, we will form a level 2 carrot at the current cell (which is [3,2]).

– ...

– step 13: This case is similar to step 5.

– ...

– step 16: Nothing happens since we do not consider diagonal neighbors. The longest link has length two in this case (2<**3**).

3.2 I/O Format For Construction of the Field

First, you will be given an integer N, indicating the size of the field. Maximum value of N is 100. Then, (as you already saw in the figure), incoming carrots will be given in form l x y. Here, l is level of the incoming carrot, x and y are the indices of the cell it will vegetate. You will not be given indices that are already full. You will be informed with EOF character when there are no more incoming carrots. For the example above, input will be like this:

```
5
1 0 1
2 2 2
2 2 3
1 0 2
1 1 2
1 3 0
1 3 1
1 3 3
1 4 3
1 3 2
2 4 2
3 1 3
2 2 2
4 2 3
4 3 4
EOF
```

Part I has an output, which is the final form of the field printed in a two-dimensional manner. For each cell, you are going to print the value of the cell and a space character after that value. At the end of each row a newline character should be printed. For the example above, the output should be as below.

```
0 0 0 0 0
0 0 0 0 0
0 0 4 0 0
0 0 4 0 0
0 0 0 4 0
```

4 Part II - Finding the Path (40 points)

After constructing the field, we are going to move the rabbit trying to maximizing the energy gain. The rabbit starts from the cell with index [0,0]. When rabbit goes to a cell, it will gain some energy based on the level of the carrot in that cell. Calculation will be made as follows:

– If a carrot is level l, it weights $100 \times l$ grams.

– If a carrot weights γ grams, the gain of energy is $0.4 \times \gamma$ calories.

You can easily see that a level 1 carrot gives 40 calories to the rabbit. Do not forget to consider calories gained from the cell [0,0].

Movement: Rabbit can only move towards south and east. You are expected to find most beneficial path and output it with the final energy.

4.1 I/O Format For Part II

As the input of this part, first you will be given an integer N which is the size of our $N \times N$ field. Then there will be N lines each consisting of N integers where each integer is separated by a space character. An example input for this part is as follows:

```
4
0 1 1 2
3 2 3 3
1 1 2 2
4 2 5 0
```

The input above corresponds to the field below:

0	1	1	2
3	2	3	3
1	1	2	2
4	2	5	0

Final energies for some possible paths are:

– E E E S S S , final energy: 360

– E E S E S S , final energy: 400

Another example:

3	1	1	2	1
3	2	3	3	1
4	1	2	2	3
4	2	5	0	4
3	2	5	1	5

Final energies for some possible paths are:

– E E E S S E S S , final energy: 920

– S S S E E S E E , final energy: 1280

You are expected to print the path with maximum energy similar to format above. You are going to print every step as E and S, put a space after each step, then print a comma(,) and a space character, and finally print "final energy: $< total\_energy\_gained >$". An end of line character after this line is also required. If there are multiple paths giving the maximum energy, printing just one of them is enough. Do not put a dash in front of the output.

A solution for the first example should print as:

S S S E E E , final energy: 600

And for the second example:

S S S E E S E E , final energy: 1280

5 Grading and Specifications

- For part I, you need to implement your solution both recursive and iterative manner. Iterative part will be 20 points and recursive one will be 40 points.
- For part II, implement your solution in recursive manner.
- Do not use static variables.
- Do not use global variables.
- Obey I/O format strictly. Otherwise you will not be able to collect points since the exam will be graded by blackbox technique.
- Since there will be two different implementations for part I, you need to submit two different files. Both of them must have a main function, since they will be executed independently. For part II you are going to submit a single file containing your solution.
- Name your files as the1_recursive.c (Recursive solution of part I), the1_iterative.c (Iterative solution of part I) and the1_part2.c (Solution of part II).
- Compress your solution files into a file named e$< student_id >$_the1.tar.gz and upload it to COW. The compressed file should not contain any directories, it should only contain the 3 files submitted. The uploaded file name for a student with id 1234567 should be e1234567_hw1.tar.gz.