COP3275 – Spring 2019

Instructor: Roozbeh Ketabi

Programming Assignment 3

1- You have to upload your codes in both Canvas and the Judge system at https://cop3275.cise.ufl.edu
2- For guide on how to access the Judge, visit Judge Access Page under Pages section in Canvas.
3- The judge will run test cases against your code and assign it a grade. You can have multiple submission for the same problem but you have to choose which one is the final (we will consider your final submission for grading). Canvas is only used for record keeping.
4- When upload on canvas, zip your programs in a zipfile with your ufid as name (nothing more, just 8 digit ufid, for instance 12345678.zip). Name your files (inside the zipped archive) p1.c, p2.c, and so on.
5- For all the problems input is read from standard input (i.e. read using scanf) and must be written to standard output (i.e. printf).
6- **Don't print extra stuff. Since the assignments are automatically graded, if the output doesn't match the expected output, you will not get the points. For instance:**
If the problem is to read a number and return its square, the expected output is a number (i.e. "printf("%d",i*i)"). If you print using something like "printf("square is %d",i*i);" you will not receive any points.
7- The assignment is due on 11:59 pm Sunday March 17th 2019. There is a 20% penalty for late submission of one day. No submission is accepted beyond that time.

Problem 1: Dense! (1 pt.)

For this problem you are given two int arrays of the same length (let's call them vectors). First compute and print (in a newline) the elementwise multiplication of them. Then determine and print (in a newline) if the given vectors are perpendicular to each other (i.e. if the inner product of them is zero, recall that inner product of vectors is the sum of elementwise multiplication). Print "perpendicular" if they are perpendicular and "not perpendicular" otherwise (**all lowercase)**. First N is given as the length of each array. Second, N numbers follow corresponding to first vector and then another N numbers are given for the second vector. N is at most 10000.

Sample input:
7
5 0 20 10 -30 5 0
8 2 -1 2 0 1 20

Sample output:
40 0 -20 20 0 5 0
not perpendicular

Problem 2: Check mate! (7 pts.)

In this problem you are given a chess board with pieces on (all formatted as characters), you have to detect whether **black** is on check, or not threatened at all.

Input is given as 8 lines of 8 characters (8 characters following by '\n' for the first 7 lines, 8 characters for the 8th line). Upper case letters stand for *white* and lower-case letters stand for *black* pieces. **Don't** assume the input is a proper/complete chess game (e.g. there might be all queens) but there always will be **only one king of each color**. A hyphen "-" stands for an empty cell. **The output must be the number of opponent (white) pieces that can capture the black king.**

You can read the piece moves on Wikipedia (https://en.wikipedia.org/wiki/Chess). In short, all pieces move and capture the same except for Pawn. Rook moves horizontally or vertically, Bishop moves diagonally, Knight has L shaped jumps, Queen moves like both Rook and Bishop, and King can move to its adjacent 8 cells only (a king cannot move to a cell where an opponent can capture it there). Pawn moves straight **toward the opponent** but can capture the piece on its immediate right or left **diagonal** cell towards the enemy (that is what matters for checking the king).

The king is in check, if it is threatened by an enemy piece (the piece can capture the king). Find and print the number of white pieces that can capture the black king (if no piece threaten the king, then print 0 which means it is not a check situation).

White: K=king, Q=queen, R= rook, B=bishop, N=knight, P=pawn

Black: k=king, q=queen, r=rook, b=bishop, n=knight, p=pawn

Sample input (the text is the input of your program; the picture corresponds to the board it is representing and is only here for demonstration):

```
---r---k
pp----Qn
--p-----
--------
--------
P-----RP
-PP---PK
--------
```

Output:
1

- Explanation: the white queen is attacking the black king.

P3: CHECKMATE! EXTRA CREDIT (3 pts.):

If the king is in check and the player doesn't have any move to get the king out of the check, it is checkmate and the player loses the match. A king may get out of the check by either moving out of the threatened cell (into a non-threatened one) or if another piece blocks the way of the opponent's piece that can capture the king (or knocks out the opponent's piece that is threatening the king). Note that the king might be threatened by two opponent pieces (more than one piece in general). In that case the only valid move is for the king to move to a non-threatened cell (and if it doesn't have such a move, then it is checkmate).

Implement checkmate functionality. Input is the same as before. Output is same as before, except when it is a checkmate, print 100. i.e. the example (with the picture above) is a case of checkmate. In this case you print 100 instead of 1. It is a checkmate because the attacking queen is being supported by a rook which means the king cannot capture the attacking queen (a king cannot move to a cell where it can be captured, in this case by the supporting rook).

- Note: this extra credit problem is substantially harder than problem 2 as you need to implement all the possible moves/captures for black player and all the captures for white player.
- In all cases if the king can move to a non-threatened cell, it is not a checkmate.
- You need to check if it is a **single check** (only one white piece can capture the king), if **any black piece can block the attacking piece or capture it** (if it can, it is not a checkmate). If the king cannot move to a non-threatened cell and if no other black piece can help, it is a checkmate and black loses the game.
- If it is a multi-check (more than one piece can capture the black king), the only possible way to get out of check is for the king to move to a cell where no other piece can capture it.