EECE 231 — Introduction to Programming Using C++ and MATLAB, Sections 1,2,6,7, and 8
Quiz II

Nov 15, 2014

- The duration of this exam is 2 and a half hours.

  Keep in mind that you need around **10 minutes** at the end of the duration of the exam to **submit** your answers. It is your responsibility to make sure your files are correctly submitted.

- The exam consists of 4 problems for 100 points (plus 14 bonus points)

- Difficulty is a subjective measure. The problems are of varying difficulty. Below is a rough ordering estimate of the problems according to "difficulty":

  - Problem 1
  - Parts (c) of Problem 2 and Part (a) of Problem 3
  - Problem 2 Parts (a), and (b)
  - Problem 3 Part (b)
  - Problem 4

  Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.

- You can use all the material available on moodle.

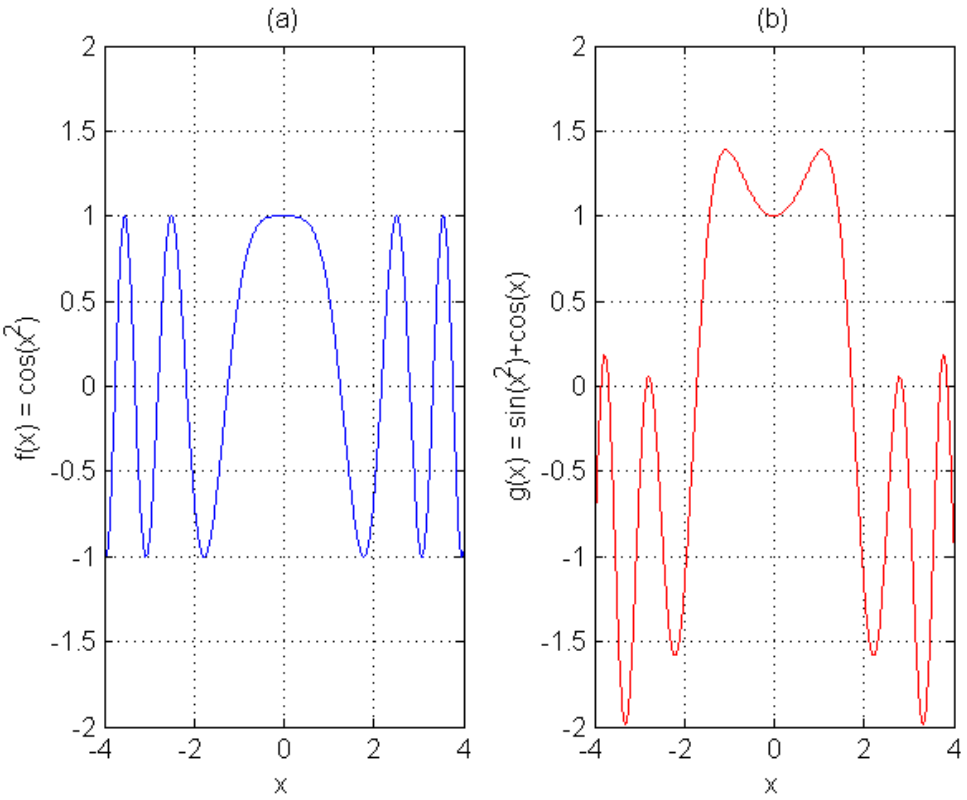
- You are **NOT** allowed to use the **web**. You are not allowed to use **USB's** or files previously stored on your machine.

- If you get caught violating the above rules or if you communicate with a person other than the exam proctors during the exam, you will immediately get zero and you will be referred to the appropriate disciplinary committee.

- Active cell phones and any other unauthorized electronic devices are absolutely not allowed in the exam rooms. They should be turned off and put away.

- Submit your solutions each part in a separate file as indicated in the booklet. Include your name and ID number in each file.

- Good luck!



Problem 1 (22 points). Graphics

Consider the functions $f(x, y) = cos(x^2)$ and $g(x) = sin(x^2) + cos(x)$. Write a Matlab script which plots them in a single figure as shown above. Show the grid and the appropriate titles and labels of the axes. To get full credit, make use of the appropriate function to produce the axis and use the point-wise operators (dot operators) as appropriate to compute the values you need for your plots. Submit your solution in a file called Prob1.m including your name and ID number in the comments.

Problem 2 (46 points). Sudoku

A Sudoku puzzle is a 9×9 matrix populated by numbers between 1 and 9 inclusive. In a valid solution of a Sudoku puzzle, each row must contain all the numbers between 1 and 9 inclusive, each column must contain all the numbers between 1 and 9 inclusive, and each of the nine 3×3 sub matrices that form the puzzle should have all the numbers between 1 and 9 inclusive.

a) **(12 points)** Write a Matlab function *checkVector* that takes a length-9 vector (row or column), returns *true* when the vector contains all the numbers between 1 and 9 inclusive, and returns $false$ otherwise.

Faster solutions are worth more points. (*Hint:* you may use an auxiliary 0/1 array $I[1 \ldots 9]$ to "book-keep" the number in the input vector. First, initialize I to zeros. Then, for number $x = v(i)$ of the input vector v, set $I(x)$ to 1 ...)

Save your function in a file called *checkVector.m*

b) **(12 points)** Write a Matlab function *checkSubMatrix* that takes a 3×3 sub-matrix, returns *true* when the sub-matrix contains all the numbers between 1 and 9 inclusive, and returns $false$ otherwise. Faster solutions are worth more points.

Save your function in a file called *checkSubMatrix.m*

c) **(22 points)** Write a Matlab script that initializes a 9×9 matrix with numbers between 1 and 9 inclusive and then uses functions *checkVector* and *checkSubMatrix* to check whether or not it is a valid Sudoku matrix. Your script should print either 'valid sudoku matrix' or 'invalid sudoku matrix'.

You may provide your solution to this part even if you have not solved parts (a) and (b) above, by writing empty functions for *checkVector* and *checkSubMatrix*. To get full credit use loops and slice indexing in your solution.

Use the valid Sudoku matrices shown in Figure (a) below to test your script. Copy the entries of the valid matrix from the file *sudokuInit.m* available with your exam. Inject one or more errors in the matrix, as shown in Figures (b) and (c), and run your script again.

Save your script in a file called *sudoku.m*

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

(a)

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

(b)

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

(c)

Detailed comments and explanations of your code are worth partial credit. Submit the files checkVector.m, checkSubMatrix.m, and sudoku.m including your name and ID number in the comments in each file.

Problem 3 (24 points). Print recursively in C++

Unlike the other problems in this exams, you are asked to solve this problem using C++ and not Matlab. The use of for or while **loops** is strictly **forbidden** in this problem.

a) **(12 points)** Write a recursive function f which given an integer n, prints the numbers $n, n-2, n-4, \ldots$ till either 0 or 1. Use the following test program.

```
f(8);
cout<<endl;
f(9);
cout<<endl;
```

You should get the following output:

```
8 6 4 2  0
9 7 5 3  1
```

b) **(12 points)** Write a recursive function g which given an integer n, prints a sequence of the following form.

- If $n = 1$, the sequence is: 1
- If $n = 2$, the sequence is: 1 2 1
- If $n = 3$, the sequence is: 1 2 1 3 1 2 1
- If $n = 4$, the sequence is: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
- If $n = 5$, the sequence is: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1 5 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1
- ...

Guess the general form of the sequence for an arbitrary value of n. Write a program to test your function.

Submit your solution in a file called Prob3.cpp including your name and ID number in the comments.

Problem 4 (8 points + 14 bonus points). Maze

A maze is an $n \times n$ matrix M of zeros and ones. A zero indicates a blocked square, and a one indicates an open square in the maze. Once in an open square (i, j), you can move to one of the four neighboring open squares: $(i-1, j)$ if $i > 1$,$(i, j-1)$ if $j > 1$, $(i+1, j)$ if $i < n$, and $(i, j+1)$ if $j < n$. The start point is at index $(1, 1)$ and the end point is at index (n, n). We will assume that $(1, 1)$ is not locked.

Write a Matlab script that initializes a maze with open and blocked squares and checks whether or not the end point is reachable from the start point to the end point by a sequence of legal moves across open squares. If a solution exists, your program should print "solution exists", otherwise, it should print "deadlocked maze".

(*Hint:* For $i = 1, \ldots, n$ and $j = 1, \ldots, n$, let $R(i, j) = 1$ if cell (i, j) is reachable from cell $(1, 1)$ by a sequence of legal moves across open squares. Otherwise, let $R(i, j) = 0$. Compute the $n \times n$ matrix R. Given R, we can check whether or not a solution exists by checking whether or not $R(n, n) = 1$. You may want to use recursion to compute R. In particular, you may want to implement the recursive function:

```
function [R] = traverseMaze(i,j,M,R)
```

Initial call:

```
R  = zeros(n,n);
R  = traverseMaze(1,1,M,R);
```

)

Examples:

```
*******************************
M =
     1     0     0     1     1
     1     1     0     0     1
     1     0     1     0     0
     1     1     0     0     1
     1     1     1     1     1
Solution exists
R =

     1     0     0     0     0
     1     1     0     0     0
     1     0     0     0     0
     1     1     0     0     1
     1     1     1     1     1
```

```
M =

     1     1     1     0     0
     1     1     1     1     0
     1     1     1     1     1
     1     0     0     0     1
     0     1     0     1     1
Solution exists

R =

     1     1     1     0     0
     1     1     1     1     0
     1     1     1     1     1
     1     0     0     0     1
     0     0     0     1     1

********************************
M =

     1     1     1     0     0
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     0     0
     1     1     0     1     1
Deadlock

R =

     1     1     1     0     0
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     0     0
     1     1     0     0     0

********************************
M =

     1     0     0     1     0
     1     1     1     0     1
     1     0     0     1     1
     0     0     1     1     0
     1     0     1     1     1
Deadlock
R =

     1     0     0     0     0
     1     1     1     0     0
     1     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
********************************
```

Any solution is worth full grade. Detailed comments and explanations of your code are also worth partial credit. Submit your solution in a file called maze.m (+ other the .m files of any other function you write for this problem), including your name and ID number in the comments.