

## Midterm Exam

COMPUTER PROGRAMMING FOR ENGINEERING AND SCIENCE  
Held on 23th June, 2016 (CSCE 155N, SUMMER 2016)

---

Name :  
Course No : CSCE155N

*Solution*

### Instructions:

1. This is open book, open note, but not open neighbor.
2. If you have a question about the meaning of an exercise, ask! Getting things wrong because of misunderstandings can be aggravating for me as well as you.
3. If answers do not fit in the space provided, use the back of a sheet and very carefully indicate and label this so I don't miss it in grading.
4. You may take the entire period.
5. Each exercise is worth 10 points, but some may take much more time to complete than others. Go through the exam and do the quick ones first!
6. There are 12 exercises. Only do 10 of them. Clearly cross off the ones you do not wish to have counted. If you fail to cross them off, the graders will instead cross off your BEST exercises!

1. (10 points) Complete the following session in Matlab.

```
>> x = [21 4 9 12 -3 8]
x =
    21     4     9    12    -3     8
```

```
>> [a b] = sort(x)
a =
    -3     4     8     9    12    21
b =
     5     2     6     3     4     1
```

```
>> x(b)
ans =
```

$2\frac{1}{2}$

-3 4 8 9 12 21

```
>> a(b)
ans =
```

$2\frac{1}{2}$

12 4 21 8 9 -3

```
>> b(b)
ans =
```

$2\frac{1}{2}$

4 2 1 6 3 5

```
>> a(b(b))
ans =
```

$2\frac{1}{2}$

9 4 -3 21 8 12

2. (10 points)

- 5 (a) Consider the following code. Picturing the array  $x$  as a multi-digit integer, what is the effect of running the code, in simple English?
- 5 (b) If the  $x(s) == 9$  and  $s > 0$  were swapped, the code might crash with an error if certain values were used for  $x$ . Explain.

```
x = [0 9 8 7 9 9 9 9];
s = length(x);
while s>0 && x(s)==9
    x(s) = 0;
    s = s - 1;
end
if s>0
    x(s) = x(s)+1;
end
disp(x)
```

a) add 1 to x

b) short circuit &amp;&amp; as is now prevents possible request for x(0)

3. (10 points) Consider the following anonymous functions. What do they return when fed the given input? Describe their function in simple English.

3/3

```
>> prune = @(e, lst) [lst<e lst>e];
>> prune(12, [-3 4 8 9 12 21])
```

-3 4 8 9 21  
removes any e's from lst

3/3

```
>> rot = @(c, lst) [lst(c+1:end) lst(1:c)];
>> rot(2, [-3 4 8 9 12 21])
```

8 9 12 21 -3 4  
rotates first c items to end of lst

3/3

```
>> mset = @(lst) [lst(lst(1:end-1) ~= lst(2:end)) lst(end)];
>> mset([3 5 5 5 7 8 8 9 9 9])
```

3 5 7 8 9  
removes duplicates from sorted list, generating a set

4. (10 points) Consider the following code.

```

count = 0;
for a = 1:n
    for b = 1:n
        count = count + 1;
    end
    for b = 1:n
        for c = 1:n
            count = count + 1;
        end
    end
    for b = 1:n
        count = count + 1;
    end
    for b = 1:n
        for c = 1:n
            for d = 1:n
                count = count+1;
            end
        end
    end
    for b = 1:n
        count = count + 1;
    end
end
end

```

Handwritten annotations in red ink to the right of the code:

- A vertical line is drawn to the right of the code, with a tick mark at the top.
- Next to the first `for b = 1:n` loop, there is a vertical line and the label  $n^2$ .
- Next to the second `for b = 1:n` loop, there is a vertical line and the label  $n^2$ .
- Next to the `for c = 1:n` loop, there is a vertical line and the label  $n^3$ .
- Next to the third `for b = 1:n` loop, there is a vertical line and the label  $n$ .
- Next to the `for c = 1:n` loop, there is a vertical line and the label  $n^2$ .
- Next to the `for d = 1:n` loop, there is a vertical line and the label  $n^3$ .
- Next to the `for b = 1:n` loop, there is a vertical line and the label  $n^4$ .
- Next to the final `for b = 1:n` loop, there is a vertical line and the label  $n^2$ .

5 (a) Give a formula (which is a function of  $n$ ) for the final value of *count*.

$$n^4 + n^3 + 3n^2$$

5 (b) Insert the minimal amount of code possible to make *count* add up to twice as much (still start at 0, of course!).

1) change `a=1:n` to `a=1:2*n`

2) add new outer loop for `x=1:2` around a loop

5. (10 points) Consider the following logic expressions.

3 1/2  
3 1/2  
3 1/2

- (a) Check the expressions in which short circuiting is enabled.
- (b) Circle the expressions in which short circuiting actually occurs.
- (c) Evaluate each expression. hint: The `fprintf` will always be `true` because a non-zero number of characters are being printed.

Here are the expressions:

- (a)  $25 > 3|fprintf('howdy') - 5$  t 1
- (b)  $-5 = 8||fprintf('howdy')$  ✓ t 1
- (c)  $45 > 84|fprintf('howdy')$  t 1
- (d)  $34 > 55||fprintf('howdy')$  ✓ t 1
- (e)  $45 > 45&fprintf('howdy')$  f 0
- (f)  $45 > 9&&fprintf('howdy')$  ✓ t 1
- (g)  $false&&fprintf('howdy') - 5$  ✓ f 0
- (h)  $2 + 16 < 6&&fprintf('howdy')$  ✓ f 0

5  
5

6. (10 points) Consider the following code. What is printed if `msg` is 'Have a nice day'? What is printed if `msg` is 'Howdy'? Describe what it does using simple English. (Note: `sprintf` returns the same string that `fprintf` would be printing. It does NOT return the length of the string. It does not actually print anything on the screen. Also, `blanks` is a function that generates spaces.)

```
m = length(sprintf('%s', msg));
fprintf('%s%s\n', blanks(40-m), msg)
```

← 40<sup>th</sup> column

Have a nice day  
Howdy

right justifies in 40 column field

- 5 7. (10 points) What is output by the following fragment of code? Are both lines of output the same? Explain.

```
n = 17;
p = mod(n,2) + 1;
f = char('even','odd')
fprintf(['%d is ', f(p,:), '\n'], n)
fprintf('%d is %s\n', n, f(p,:))
```

*builds format with correct parity*

*17 is odd*  
*17 is odd*

- 10 8. (10 points) What does the following code print when it is invoked from the command line with *funone*. Be reasonably careful with the spacing. Note that *blanks* is a built-in function which generates a string with the specified number of spaces.

```
function funone()
    a = 3;
    indent(a, 'Getting started');
    funtwo(a+3);
    indent(a, 'At the halfway mark');
    funthree(a+3);
    indent(a, 'Reached the finish line');
end

function funtwo(b)
    indent(b, 'intwo');
    if b <= 12
        funthree(b+3);
    end
    indent(b, 'outtwo');
end

function funthree(a)
    indent(a, 'inthree')
    if a <= 12
        funtwo(a+3);
    end
    indent(a, 'outthree')
end

function indent(b, msg)
    fprintf('%d%s %s\n', b, blanks(b), msg);
end
```

*3 Getting started*  
*6 in two*  
*9 in three*  
*12 in two*  
*12 out two*  
*9 out three*  
*6 out two*  
*3 At the halfway mark*  
*6 in three*  
*9 in two*  
*12 in three*  
*12 out three*  
*9 out two*  
*6 out three*  
*3 Reached the finish line*

Held: Thursday, June 23, 2016

10

9. (10 points) What is output by the following fragment of code?

```
fprintf('start a\n')
for a = 2:-2:-2
    fprintf('start b\n')
    for b = a:2
        fprintf('start c\n')
        for c = b:-2:a
            fprintf('%1d %1d %1d\n', a, b, c)
        end
        fprintf('exit c\n')
    end
    fprintf('exit b\n')
end
fprintf('exit a\n')
```

Handwritten output for question 9:

```

start a
start b
start c
2 2 2
exit c
exit b

start b
start c
0 0 0
exit c
start c
0 1 1
exit c
start c
0 2 2
0 2 0
exit c
exit b

start b
start c
-2 -2 -2
exit c
start c
-2 -1 -1
exit c
start c
-2 0 0
-2 0 -2
exit c

```

10

10. (10 points) Consider the following function definition. Write an expression that uses it (and only it - so no operators!) to calculate the volume of ice cream filling a conical ice cream cone topped with a half sphere lump of ice cream, given the radius of the top of the cone as  $r$  and height as  $h$ .

```
function x = mOp(a, op, b)
switch op
case '+'
    x = a + b;
case '-'
    x = a - b;
case '*'
    x = a * b;
case '/'
    x = a / b;
case '^'
    x = a ^ b;
otherwise
    x = 0;
end
```

Handwritten formulas for question 10:

$$((\pi r^2 h) / 3) + 2\pi r^3 / 3 \quad (\text{may vary})$$

$$(\pi r^2 h + 2\pi r^3) / 3$$

$$\pi (r^2 h + 2r^3) / 3$$

$$mOp(mOp(\pi, '^', mOp(r, '^', 2)), '*', mOp(h, '+', mOp(2, '^', r))) / 3$$

Handwritten final expression for question 10:

$$mOp(mOp(mOp(mOp(\pi, '^', mOp(r, '^', 2)), '*', mOp(h, '+', mOp(2, '^', r))), '/', 3))$$

11. (10 points) Cross off any redundant (unnneeded) portions of the code. (The resulting code should always yield the same results, regardless of the input which COULD be outside the specified range.)

```
grade = input('Enter the grade from 0 to 100: ');
```

```
1 if grade > 80 && grade <= 90
```

```
    disp('Not a bad BB')
```

```
1 elseif grade < 0 || grade > 100
```

```
    disp('Invalid input')
```

```
2 elseif grade <= 50 && grade >= 0
```

```
    disp('Sorry - you blew it')
```

```
1 elseif grade > 60 && grade <= 70
```

```
    disp('Discouraging D')
```

```
1 elseif grade > 50 && grade <= 60
```

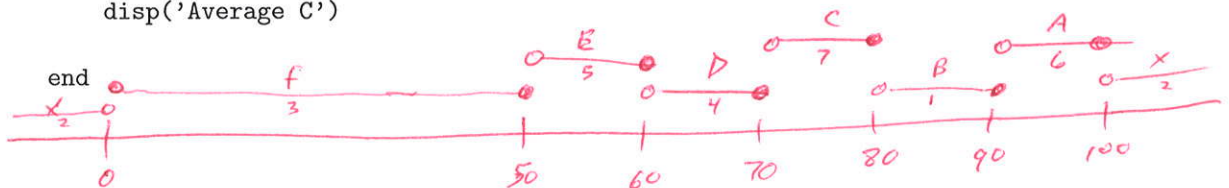
```
    disp('An E for effort is all you get')
```

```
1 elseif grade > 90 && grade <= 100
```

```
    disp('Nice A')
```

```
2 elseif grade > 70 && grade <= 80
```

```
    disp('Average C')
```



- 10 12. (10 points) Convert the following nested *for* loop into an equivalent *while* loop.

```
for ii = -10:2:10
    for jj = 0:5
        printf('%d %d\n', ii, jj)
    end
end
```

```
ii = -10
while ii <= 10
    jj = 0
    while jj <= 5
        printf('...')
        jj = jj + 1
    end
    ii = ii + 2
end
```



Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
11	10	
12	10	
Total:	100	