

MCS-178 Final Exam

Serial #:

This exam is closed-book and mostly closed-notes. You may, however, use a single 8 1/2 by 11 sheet of paper with *hand-written* notes for reference. (Both sides of the sheet are OK.)

Please write your name only on this page. Be sure to look at all problems before deciding which one to do first. Some problems are easier than others, so plan your time accordingly. You have 120 minutes to work.

Write the answer to each problem on the page on which that problem appears. You may also attach additional paper, which should be labeled with your test number and the problem number.

Printed name: _____

On my honor, I pledge that I have not given, received, nor tolerated others' use of unauthorized aid in completing this work.

Signature for above honor pledge: _____

Problem	Page	Possible	Score
1	2	13	
2	3	13	
3	4	14	
4	7	12	
5	8	12	
6	9	12	
7	10	12	
8	11	12	
Total		100	

1. [**13 Points**] Write *either* a memoized *or* a dynamic programming version of the following tree-recursive procedure. You may use any kind of data structure you like for the table and may define it either as a global variable or using nesting, so long as your solution is correct.

```
def f(n): # n must be a nonnegative integer
    if n==0:
        return 0
    elif n==1:
        return 10
    elif n==2:
        return 20
    else:
        return f(n-3) + 3*f(n-2) + f(n-1)
```

2. [13 Points] Consider the following EBNF grammar:

$$\langle s \rangle \longrightarrow \langle s \rangle \langle s \rangle \langle s \rangle$$

| **x**

$$\langle t \rangle \longrightarrow \mathbf{x^+ x x}$$

- (a) Give four strings that are in the language of $\langle s \rangle$.
- (b) What string is in the language of $\langle s \rangle$ but not the language of $\langle t \rangle$?
- (c) Give an example of a string in the language of $\langle t \rangle$ that is not in the language of $\langle s \rangle$.
- (d) Give an EBNF grammar for $\langle u \rangle$ so that its language contains all strings that start with zero or more copies of **a**, then have a single **m**, and then have the same number of copies of **z** as there were of **a**. (Example strings would be **m**, **a m z**, and **a a m z z**.)

3. [14 Points] This problem concerns the following example code from early November:

```
class Point(object):
    """represents a point in 2-D space"""
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __add__(self, other):
        return Point(self.x + other.x, self.y + other.y)

class Rectangle(object):
    """represent a rectangle.
    attributes: width, height, corner (lower left), color.
    """
    def __init__(self, corner, height, width, color):
        self.corner = corner
        self.height = height
        self.width = width
        self.color = color

    def draw(self, canvas):
        """Draws self on canvas."""
        upperX = self.corner.x + self.width
        upperY = self.corner.y + self.height
        bbox = [[self.corner.x, self.corner.y], [upperX, upperY]]
        canvas.rectangle(bbox, outline='black', width=2, fill=self.color)

    def translate(self, displacement):
        self.corner = self.corner + displacement

class Circle(object):
    """represent a circle.
    attributes: radius, center, color.
    """
    def __init__(self, radius, center, color):
        self.radius = radius
        self.center = center
        self.color = color

    def draw(self, canvas):
        """Draws self on canvas."""
        canvas.circle([self.center.x, self.center.y], self.radius,
                      outline=None, fill=self.color)

    def translate(self, displacement):
        self.center = self.center + displacement
```

```

class Polygon(object):
    """represent a polygon.
       attributes: points, color.
    """
    def __init__(self, points, color):
        self.points = points
        self.color = color

    def draw(self, canvas):
        """Draws self on canvas."""
        canvas.polygon([[point.x, point.y] for point in self.points],
                       outline=None, fill=self.color)

    def translate(self, displacement):
        self.points = [point + displacement for point in self.points]

class Composite(object):
    def __init__(self, components):
        self.components = components

    def draw(self, canvas):
        for i in self.components:
            i.draw(canvas)

    def translate(self, displacement):
        for i in self.components:
            i.translate(displacement)

```

Suppose we now want to add a method named `minX` to each kind of image (`Rectangle`, `Circle`, `Polygon`, and `Composite`) such that if `i` is an image, `i.minX()` will return the minimum x coordinate of any portion of the image, that is, how far to the left the image extends. The method in the `Rectangle` class would be

```

def minX(self):
    return self.corner.x

```

and the method in the `Polygon` class would be

```

def minX(self):
    return min([point.x for point in self.points])

```

(This latter method makes use of the predefined procedure `min`, which can be given a list of numbers and will then return the least of those numbers.)

(a) How should `minX` be defined within the `Circle` class?

(b) How should `minX` be defined within the `Composite` class?

4. [12 Points] Recall our calculator program:

```
def calc():
    g = Gui()
    display = g.en()
    def doIt():
        expression = display.get()
        result = str(eval(expression))
        clear()
        display.insert(0, result)
    def clear():
        display.delete(0, END)
    g.gr(4, rweights=[1]*5, cweights=[1]*4)
    for key in [7,8,9,'+',4,5,6,'-',1,2,3,'*',0,'.', 'E', '/', '(', ')']:
        key = str(key)
        g.bu(text=key, command=Callable(display.insert, INSERT, key))
    g.bu(text="C", command=clear)
    g.bu(text="=", command=doIt)
    g.endgr()
    g.mainloop()
```

Modify this program so that there is one additional button, extending across the full width of the calculator below the main grid of buttons. This new button should be labeled `parenthesize` and when pressed should modify the display to have a left parenthesis at the beginning and a right parenthesis at the end. You can just show your new code and use an arrow to indicate where within the existing code it should be inserted.

5. [**12 Points**] The *Joy of Cooking* suggests that to figure out how many people a turkey will serve, you should allow $3/4$ of a pound per person for turkeys up to 12 pounds in weight, but $1/2$ pound per person for larger turkeys. Write a SLIM program that reads in a turkey weight in pounds and writes out the number of people the turkey will serve. You should assume that SLIM can only operate on integers. If the exact number of persons isn't an integer, the answer should be rounded down rather than up; we don't want to risk having too little turkey.

6. [12 Points] Any positive integer can be expressed as a power of two times an odd number; for example, $40 = 2^3 \cdot 5$. The following SLIM program reads in a number and writes out the corresponding exponent of two and odd number. For example, if it reads in 40, it will write out 3 and 5:

```

        allocate-registers loop, even, one, two, n, exponent, r
        li loop, Lloop
        li even, Leven
        li one, 1
        li two, 2
        read n
        li exponent, 0
Lloop:
        rem r, n, two
        jeqz r, even
        write exponent
        write n
        halt
Leven:
        add exponent, exponent, one
        div n, n, two
        j loop

```

- If this program is run and an odd number is input, how many instructions will the computer execute?
- If the program is run and 10 is input, how many instructions will the computer execute?
- If the program is run and 40 is input, how many instructions will the computer execute?
- If the program is run and 0 is input, what will the computer do?
- Suppose the computer is broken such that it cannot execute j instructions. To work around this problem, modify the program so it doesn't use j .
- How many instructions longer is your new program than the original?
- How many more registers does your new program use than the original?
- How many more instructions does your program execute than the original, when each is given 40 as its input?

7. [12 Points] The following SLIM program has four instructions missing, indicated by blank lines (_____). The goal of the program is to read in a number, n , and then use a recursive process to print out the digits of that number, starting with the leftmost one. For example, if the program reads in 314, it should write out 3, then 1, and then 4.

```

    allocate-registers printDigits, baseCase, ten, one
    allocate-registers sp, n, continuation, hasMultipleDigits
    li printDigits, LprintDigits
    li baseCase, LbaseCase
    li ten, 10
    li one, 1
    li sp, 0
    read n
    li continuation, LafterTopLevel
    j printDigits
LafterTopLevel:
    halt

LprintDigits:
    ;; print out the digits of n, from leftmost to rightmost
    ;; and then jump to wherever the continuation register specifies
    sge hasMultipleDigits, n, ten
    jeqz hasMultipleDigits, baseCase
    st n, sp
    add sp, sp, one
    st continuation, sp
    add sp, sp, one
    quo n, n, ten
    li continuation, LafterRecursiveCall
    j printDigits
LafterRecursiveCall:
    _____
    _____
    _____
    _____
    rem n, n, ten
LbaseCase:
    write n
    j continuation

```

- When n is 3, how many data memory locations does the program store into?
- What are the last two instructions this program should execute? (Hint: they are not adjacent.)
- Fill in the four blanks.

8. [**12 Points**] Write a program using the `Gui` class that contains three components: a text entry field, a button labeled **Reverse**, and a text label that is initially empty. Whenever the button is pressed, the label should be reconfigured to contain a reversed version of the string from the entry field. (If you'd rather, you can do some other string transformation, like converting to pig latin or all uppercase.)