

MCS-178 Intra-term Exam 2

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 50 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	17	
2	3	16	
3	4	18	
4	5	16	
5	6	17	
6	7	16	
Total		100	

1. [17 Points] The following class defines a new kind of thing, the thingamajig:

```
class Thingamajig:
    def __init__(self):
        self.contents = [None, None]
        self.place = 0

    def insert(self, value):
        self.contents[self.place] = value
        self.place = (self.place + 1) % 2

    def retrieve(self):
        return self.contents[self.place]
```

- (a) If someone creates a new thingamajig using `t = Thingamajig()`, what attributes does `t` have, and what are the values of those attributes?
- (b) If someone now executes `t.insert(17)`, what do the attribute values become? As a reminder, the `%` symbol represents the remainder operation.
- (c) Continuing, suppose the user then executes `t.insert(42)`. What do the attribute values become?
- (d) At this point in the example, what would the value of `t.retrieve()` be?
- (e) More generally, if some insertions and retrievals are done on a newly created thingamajig, how could you predict what each retrieval was going to retrieve? Your explanation shouldn't talk about such internal matters as attributes, lists, and positions within lists. Instead, you should talk about how insertion and retrieval relate. Be sure your explanation would work no matter what order the insertions and retrievals take place in.

2. [16 Points] The following is the action table from our shift/reduce parser (before you modified it):

stack top	next token				
	\$	op	num	()
\$	error	error	shift	shift	error
op	error	error	shift	shift	error
num	reduce, accept, or error	shift or reduce	error	error	shift, reduce, or error
(error	error	shift	shift	error
)	reduce	reduce	error	error	reduce

Suppose we want to allow multiplication to be implied by juxtaposition, so that $2 + 3 5$ would evaluate to 17, $2(3+4)$ would evaluate to 14, $(1+2)(3+4)$ to 21, and $(1+2)3$ to 9, for example. This should obey the same grouping rules as explicit multiplication does. In particular, $8/4 3$ would evaluate to 6, just like $8/4*3$ does.

- Two of the “error” entries in the table need to be changed to “reduce”. Mark those changes.
- Two other “error” entries need to be changed to “shift or reduce”. Mark those changes.
- For the two new “shift or reduce” entries, the decision whether to shift or reduce would be based on the item on the stack immediately below the top item. What items in that position would make it appropriate to reduce?
- There is a new kind of reduction. How many items does it pop off the top of the stack? What type of items will they be? How should the value be calculated to push back on the stack?

3. [18 Points] Consider the following EBNF grammar:

$$\langle s \rangle \longrightarrow \begin{array}{c} \mathbf{a a} \langle s \rangle \mathbf{b} \\ | \mathbf{c} \end{array}$$

$$\langle t \rangle \longrightarrow \mathbf{a^+ c b^+}$$

- (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 of odd length that alternate between \mathbf{a} and \mathbf{b} , starting with an \mathbf{a} . (Example strings would be \mathbf{a} , $\mathbf{a b a}$, and $\mathbf{a b a b a}$.) You might find it helpful to define at least one additional nonterminal in your grammar beyond just $\langle u \rangle$.

4. [**16 Points**] Define a new class, **Brat**, which inherits from the **AutoPerson** class in the Land of Gack. A **Brat** is distinguished by the following features:
- (a) His **haveFit** method is different from that of a normal **Person**. Instead of saying “Yaaaah! I am upset!”, he says “You call this fun?”.
 - (b) Each time he acts, he first has a fit, but then goes on to act in the same way as he would if he were a normal **AutoPerson**. (You should not achieve this latter objective by duplicating code from **AutoPerson**.)

5. [17 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()
```

Suppose we add the following variable definition within the `calc` procedure:

```
memory = [""]
```

The idea is that we can use `memory[0]` as a memory storage location initially containing an empty string. (For now, there will only be one location.) Show the changes that would be necessary to add one more row of buttons, containing the following four buttons. You can mark changes on the existing code and use an arrow to indicate where to insert new code that you write. You can write the new code on a separate sheet of paper, labeled with your test's serial number and the problem number.

- A button labeled MC that clears the memory location back to the empty string.
- A button labeled MS that first does the same thing as the = button, so as to reduce whatever is in the display to a single number, and then stores that value into the memory location, leaving it also in the display.
- A button labeled MI that inserts the contents of the memory location into the display, in addition to whatever is already there.
- A button labeled MR that puts the contents of the memory location in the display, replacing whatever might have been in the display before.

6. [16 Points] Recall our Composite class:

```
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 that the other kinds of images, such as Circles and Polygons, have been upgraded to support one more method, `estimatedDrawingTime`. For example, if `c` is a Circle, the evaluating `c.estimatedDrawingTime()` would return an estimate of how many microseconds would be needed to draw the circle on the screen. Show how to provide this same new feature in the Composite class.