CS 345 - Programming Languages
Fall 2010

Homework #7

Due: 2pm CST (in class), December 2, 2010

YOUR NAME: _____

Collaboration policy

No collaboration is permitted on this assignment. Any cheating (*e.g.*, submitting another person's work as your own, or permitting your work to be copied) will automatically result in a failing grade. The Computer Science Department Code of Conduct can be found at `http://www.cs.utexas.edu/academics/conduct/`.

Late submission policy

This homework is due at the **beginning of class** on **December 2**. All late submissions will be subject to the following policy.

You start the semester with a credit of 3 late days. For the purpose of counting late days, a "day" is 24 hours starting at 2pm on the assignment's due date. Partial days are rounded up to the next full day. You are free to divide your late days among the take-home assignments any way you want: submit four assignments 1 day late, submit one assignment 3 days late, *etc.* After your 3 days are used up, no late submissions will be accepted and you will automatically receive 0 points for each late assignment.

You may submit late assignments to Vitaly Shmatikov (CSA 1.114—slide under the door if the office is locked). **If you are submitting late, please indicate how many late days you are using.**

Write the number of late days you are using: _____

Homework #7 (35 points)

Problem 1 (5 points)

The following webpage contains a "cross-site scripting" vulnerability:

`http://www.cs.utexas.edu/~shmat/courses/cs345/vulnerable.html`

Imagine that you want to trick your fellow CS 345 student into believing that the course has a different collaboration policy.

Craft a URL that will cause the above website to display an alternate collaboration policy. You may assume that your victim will visit your webpage and click on a special link that you have created. This link must point to the above page. When the victim clicks on it, he should see the above page, but the displayed collaboration policy should be different.

IMPORTANT: Do not point your link to a modified copy of the above page. Your link should point to `http://www.cs.utexas.edu/~shmat/courses/cs345/vulnerable.html`, or you will get no credit for this problem.

Problem 2

Consider the following PHP script for logging into a website:

```
$username = addslashes($_GET[user]);
$password = addslashes($_GET[pwd]);
$sql = "SELECT * FROM usertable
        WHERE username= '$username' AND password = '$password' ";
$result = $db->query($sql);
if ($result->num_rows > 0) { /* successful login */ }
else                       { /* login failed */ }
```

Problem 2a (2 points)

What does the PHP function `addslashes` do? What is the purpose of calling it in the above code?

Problem 2b (3 points)

In Chinese, Korean, and Japanese unicode character sets, some characters are encoded as single bytes, while others are double bytes. For example, the database interprets `0x5C` as \, `0x27` as ', `0x5C27` as \', but `0xBF5C` is interpreted as a single Chinese character.

Give an example of a username that will successfully subvert the above authentication code.

Problem 2c (2 points)

How should `addslashes` be implemented to prevent SQL injection attacks?

Problem 3

In Simula, a class is a procedure that returns a pointer to its activation record. Simula prefixed classes are a precursor to C++ derived classes, providing a form of inheritance.

This question asks about how inheritance might work in an early version of Simula, assuming that the standard static scoping mechanism associated with activation records is used to link the derived class part of an object with the base class part of the object.

Sample `Point` and `ColorPt` classes are given in the textbook (Section 11.2). For the purpose of this problem, assume that if `cp` is a `ColorPt` object, consisting of a `Point` activation record followed by a `ColorPt` activation record, the access link of the parent class (`Point`) activation record points to the activation record of the scope in which the class declaration occurs, and the access link of the child class (`ColorPt`) activation record points to the activation record of the parent class.

Problem 3a (3 points)

Fill in the missing information in the activation records (depicted below), which are created by executing the following code:

```
ref(Point) r;
ref(ColorPt) cp;
r :- new Point(2.7, 4.2);
cp :- new ColorPt(3.6, 4.9, red);
cp.distance(r);
```

Remember that function values are represented by closures, and that a closure is a pair consisting of an environment (pointer to an activation record) and compiled code. In this drawing, a bullet (•) indicates that a pointer should be drawn from this slot to the appropriate closure. Since the pointers to activation records cross and could become difficult to read, each activation record is numbered at the far left. In each activation record, place the number of the activation record of the statically enclosing scope in the slot labeled "access link." The first two are done for you. Also use activation record numbers for the environment pointer part of each closure pair. Write the values of local variables and function parameters directly in the activation records.

Activation records

(0)	r	(1)
	cp	(3)
(1) Point(. . .)	access link	(0)
	x	
	y	
	equals	•
	distance	•
(2) Point part of cp	access link	(0)
	x	
	y	
	equals	•
	distance	•
(3) ColorPt(. . .)	access link	( )
	c	
	equals	•
(4) cp.distance(r)	access link	( )
	q	(r)

Closures

⟨ ( ), • ⟩

⟨ ( ), • ⟩

⟨ ( ), • ⟩

⟨ ( ), • ⟩

Compiled code

code for equals

code for distance

code for cpt equals

Problem 3b (2 points)

The body of distance contains the expression

```
sqrt((x - q.x)**2 + (y - q.y)**2)
```

which compares the coordinates of the point containing this `distance` procedure to the coordinate of the point q passed as an argument. Explain how the value of x is found when `cp.distance(r)` is executed, mentioning specific links in your diagram. What value of x is used?

Problem 3c (2 points)

The above illustration shows that a reference `cp` to a colored point object points to the `ColorPt` part of the object. Assuming this implementation, explain how the expression `cp.x` can be evaluated. Explain the steps used to find the right x value on the stack, starting by following the pointer `cp` to activation record (3).

Problem 3d (2 points)

Explain why the call `cp.distance(r)` only needs access to the `Point` part of `cp` and not the `ColorPt` part of `cp`.

Problem 4

We can compare Smalltalk interfaces to classes using protocols, which are lists of operation names (selectors). When a selector allows parameters, as in `at: put:`, the selector name includes the colons but not the spaces. More specifically, if `dict` is an updatable collection object, such as a dictionary, then we could send dict a message by writing `dict at:'cross' put:'angry'`. (This makes our dictionary definition of "cross" the single word "angry.") The protocol for updatable collections will therefore contain the seven-character selector name `at:put:` . Here are some example protocols:

```
              map : {fnumElems, contains:, at:put:, get:, remove:}
              set : {fnumElems, contains:, add:, remove:}
         multiset : {fnumElems, contains:, count:, add:, remove:}
            array : {fnumElems, at:put:, get:}
simple_collection : {fnumElems}
```

Briefly, a `map` represents a function from a set of integer keys to a set of values. It can be sent the message `isEmpty` to test whether its domain is empty, returning `true` if empty and `false` otherwise; `contains:` tests whether a given key is part of the domain; `at:put:` adds a key/value mapping to the map; `get:` fetches the value associated with the given key, and remove: removes the mapping from the given key. A `set` object represents a set of integers, where the `contains:`, `add:` and `remove:` methods test membership, add an element, and remove an element respectively. Multisets are like sets except they may contain multiple copies of any given element, which we can quantify using the `count:` method. Arrays are objects where the `at:put:` method is used to update elements of the array, and the `get:` method is used to retrieve elements of the array. The `simple_collection` class just collects methods that are common to all the other classes. We say that the protocol for `A` *conforms* to the protocol for `B` if the set of `A` selector names contains the set of `B` selector names.

Problem 4a (3 points)

Draw a diagram of these classes, ordered by protocol conformance. You should end up with a graph that looks like William Cook's drawing shown in the textbook, without the dotted arrows showing inheritance.

Problem 4b (2 points)

Choose two classes, A and B, from your diagram from part (a) such that B conforms to the protocol for A. Describe briefly, in words, a way of implementing each class so that the implementation of B inherits from the implementation of A.

Problem 4c (2 points)

Choose two classes, A and B, from your diagram from part (a) such that neither conforms to the protocol of the other. Describe briefly, in words, a way of implementing each class so that the implementation of A inherits from the implementation of B.

Problem 4d (2 points)

Choose two classes, A and B, from your diagram from part (a) such that B conforms to the protocol for A. Describe briefly, in words, a way of implementing each class so that the implementation of A inherits from the implementation of B.

Problem 5

Smalltalk has a mechanism for "undefining" a method. Specifically, if a class A has method m, then a programmer may cancel m in subclass B by writing:

```
m:
  self shouldNotImplement
```

With this declaration in subclass B, any invocation of m on a B object will result in a special error indicating that the method should not be used.

Problem 5a (2 points)

What effect does this feature of Smalltalk have on the relationship between inheritance and subtyping?

Problem 5b (3 points)

Suppose class A has methods m and n, and method m is canceled in subclass B. Method n is inherited and not changed, but method n sends the message m to self.

What do you think happens if a B object b is sent a message n? There are two possible outcomes—what are they? Which one do you think the designers of Smalltalk would have chosen and why?