CMPSC 461: Programming Language Concepts
Assignment 7 Solution

Problem 1 [8pt] Prove that the following two Hoare triples are valid. (Hint: in predicate logic $P_1 \Rightarrow P_2$ is equivalent to $\neg P_1 \vee P_2$).

a) (4pt)

```
{$x = 1$}
y := x + 3;
y := y * 4;
{$y > 11$}
```

Solution:

$$\begin{aligned} wp(\texttt{y:=x+3;y:=y*4}, y > 11) &= wp(\texttt{y:=x+3}, wp(\texttt{y:=y*4}, y > 11)) \\ &= wp(\texttt{y:=x+3}, y * 4 > 11) \\ &= (x+3) * 4 > 11 \end{aligned}$$

Moreover, the precondition $x = 1$ implies the weakest precondition $(x + 3) * 4 > 11$.

b) (4pt)

```
{$y < 6$}
if (y>3) x := y-2;
else     x := 7-y;
{$x > 1$}
```

Solution:

$$\begin{aligned} wp(\textbf{if}\ \texttt{(y>3) x:=y-2; else x:=7-y}, x > 1) &= ((y > 3) \Rightarrow wp(\texttt{x:=y-2}, x > 1)) \wedge \\ &\quad ((y \leq 3) \Rightarrow wp(\texttt{x:=7-y}, x > 1)) \\ &= ((y > 3) \Rightarrow y - 2 > 1) \wedge ((y \leq 3) \Rightarrow 7 - y > 1) \\ &= ((y > 3) \Rightarrow y > 3) \wedge ((y \leq 3) \Rightarrow 6 > y) \\ &= \texttt{true} \wedge \texttt{true} = \texttt{true} \end{aligned}$$

Moreover, the precondition $y < 6$ implies the weakest precondition `true`.

Problem 2 [14pt] Below is a Hoare triple that includes pseudo code which computes the product c of two integers a and b.

```
{$b \geq 0$}
c := 0;
n := b;
while (n>0) {
  c := c + a;
  n := n - 1;
}
{$c = a \times b$}
```

a) (4pt) Complete the following Hoare triple by writing down the strongest postcondition for the first two statements in this program fragment.

```
{$b \geq 0$}
c := 0;
n := b;
{                          }
```

Solution:

$$\begin{aligned} sp(\texttt{c:=0;n:=b}, b \geq 0) &= sp(\texttt{n:=b}, sp(\texttt{c:=0}, b \geq 0)) \\ &= sp(\texttt{n:=b}, \exists v.\ c = 0 \wedge b \geq 0) \\ &= sp(\texttt{n:=b}, c = 0 \wedge b \geq 0) \\ &= (\exists v.\ n = b \wedge c = 0 \wedge b \geq 0) \\ &= (n = b \wedge c = 0 \wedge b \geq 0) \end{aligned}$$

b) (4pt) Write down a loop invariant for proving the correctness of this program. That is, write down an invariant that is 1) true before the first execution of the loop, 2) true before execution of every successive repetition of the loop, and 3) strong enough to prove the correctness of the code (i.e., `c=a×b` is true after the loop).

Solution:
The invariant is $n \geq 0 \wedge c = (b - n) \times a$.

c) (6pt) Briefly explain why the invariant you developed satisfies all three conditions stated in the previous problem. You might need to use your answer of 2 a) for this question.

Solution:
We use $\mathcal{I}$ to represent the loop invariant.

Condition 1): easy to check that $n = b \wedge c = 0 \wedge b \geq 0$ implies $\mathcal{I}$.

Condition 2): we need to check the following Hoare triple is true:

```
{n ≥ 0 ∧ I}
c = c + a;
n = n - 1;
{I}
```

Based on the postcondition $\mathcal{I}$, we can compute that the weakest precondition is

$$n - 1 \geq 0 \wedge c + a = (b - n + 1) \times a$$

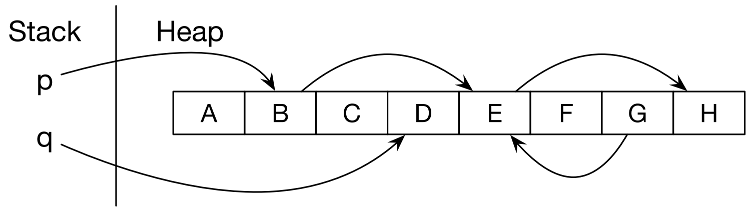
which can be simplified as

$$n \geq 1 \wedge c = (b - n) \times a$$

So the Hoare triple above is correct since $n > 0 \wedge \mathcal{I}$ implies $n \geq 1 \wedge c = (b - n) \times a$ (note that n is an integer, so $n > 0$ implies $n \geq 1$).

Condition 3): Easy to check that $n \leq 0 \wedge \mathcal{I}$ implies the postcondition $c = a \times b$.

Problem 3 [16pt] Consider the heap state as shown below before garbage collection:



a) (4pt) Consider the Mark-and-Sweep algorithm. What objects remain on the heap after collection?

Solution:
Objects B,D,E,H remain on the heap after collection.

b) (12pt) Consider algorithms Mark-and-Sweep, Mark-and-Compact, Stop-and-Copy. For each of them, write down the objects being visited during the collection in sequence. Assume 1) reachability analysis uses depth-first search, which will skip objects that have already been visited, 2) search starts from p, and then q, 3) when the entire heap is traversed, objects are visited from left to right. You don't need to write down the newly created object copies, if any.

Solution:
Mark-and-Sweep: B, E, H, D (Mark), A, B, C, D, E, F, G, H (Sweep).

Mark-and-Compact: B, E, H, D (Mark), A, B, C, D, E, F, G, H (Compute new addresses), A, B, C, D, E, F, G, H (Move objects to new address).

Stop-and-Copy: B, E, H, D.

Problem 4 [12pt] Consider the following Java classes:

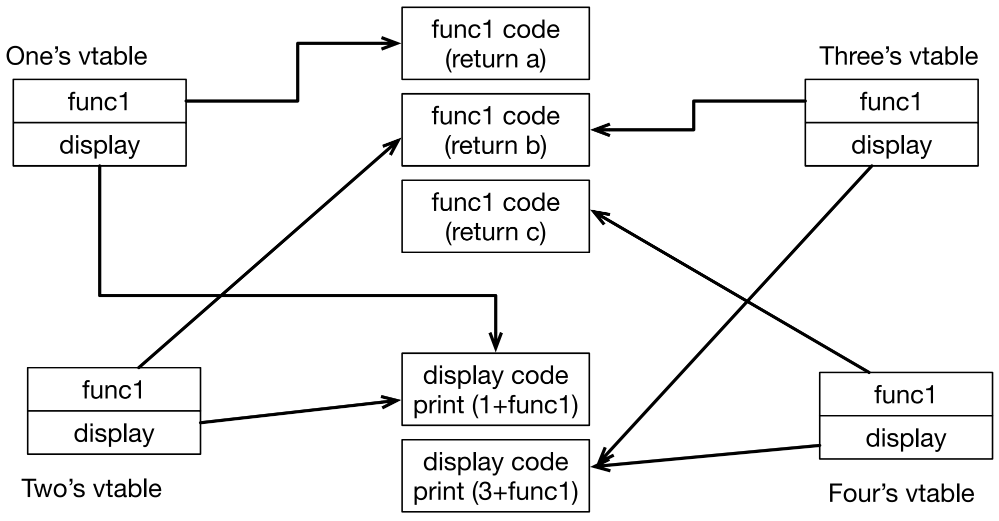
```
class One {
    public char func1 () { return 'a';}
    public void display () { System.out.println( 1 + func1()); }
}

class Two extends One {
    public char func1() {return 'b';}
}

class Three extends Two {
    public void display () { System.out.println( 3 + func1()); }
}

class Four extends Three {
    public char func1() {return 'c';}
}
```

a) (6pt) Draw the virtual tables of class One, Two, Three and Four.



b) (6pt) Consider the following polymorphic function:
`void func(Two two) { two.display(); }`
What are the outputs of the following code? (Hint: Java uses dynamic dispatching.) What would be the outputs if Java used static dispatching?

```
Three three = new Three();
Four four = new Four();
func(three);
func(four);
```

Solution:
The outputs are “3b” and “3c”.

The outputs of static dispatching are “1b” and “1b”.