Final Exam – Solutions

Question 1:

If the type checker tries to check the statement C().m(5,6) then it will have to consult the symbol table of class C to get the type of the method m.

Assuming that we have a 2 pass compiler, the symbol table of class C should at this time look like:

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.x</td>
<td>Int</td>
</tr>
<tr>
<td>C.y</td>
<td>Int</td>
</tr>
<tr>
<td>C.m.num</td>
<td>Int</td>
</tr>
<tr>
<td>C.m.a</td>
<td>Int</td>
</tr>
<tr>
<td>C.m.i</td>
<td>Int</td>
</tr>
<tr>
<td>C.m.j</td>
<td>Int</td>
</tr>
<tr>
<td>Methods</td>
<td></td>
</tr>
<tr>
<td>C.m</td>
<td>Int</td>
</tr>
</tbody>
</table>

Then the type check would just require access to the method field C.m

Question 2:

Heap layout of the C object where T1 and T2 are addresses allocated by HALLOCATE

```
    T1+0
    └── Methods pointer
         └── Value of X
             └── Value of Y
         └── T2+0
             └── Method m()
```

Piglet translation of the code:

PRINT CALL
BEGIN
  MOVE TEMP 24
    BEGIN
      MOVE TEMP 25 HALLOCATE 4
      MOVE TEMP 26 HALLOCATE 16
      HSTORE TEMP 25 0 C_m
      MOVE TEMP 27 4
      L0 CJUMP LT TEMP 27 16 L1
      HSTORE PLUS TEMP 26 TEMP 27 0 0
      MOVE TEMP 27 PLUS TEMP 27 4
      JUMP L0
      L1 HSTORE TEMP 26 0 TEMP 25
    RETURN
    TEMP 26
  END
HLOAD TEMP 22 TEMP 24 0
  MOVE TEMP 23 PLUS TEMP 22 0
RETURN
TEMP 23
END
(TEMP 24 5 6 )
END
Question 3:
Let int[] ar be the first field declaration in a class C. Then the offset of the pointer to the array structure will be 4 (from the beginning of the class structure)
In other words if the class C heap layout starts at TEMP 0 then the array pointer will be at TEMP 0 + 4

Assuming HALLOCATE returned address T2 we do the following allocation for the array:

If we want to access the element a[3] and suppose that the pointer to the Array layout is stored in class variable TEMP 20 then the Piglet code for that would look like:

```
MOVE TEMP 21
BEGIN
  HLOAD TEMP 32 PLUS TEMP 20 PLUS BEGIN
    MOVE TEMP 30 TIMES 3 4
    HLOAD TEMP 31 TEMP 20 0
    CJUMP MINUS 1 LT TEMP 30 TEMP 31 L4
    ERROR
    L4    NOOP
    RETURN
    TEMP 30
  END
  4 0
  RETURN
  TEMP 32
END
```

RETURN TEMP 21
Question 4:

Using the algorithm 10.4 in the book we construct the following table

<table>
<thead>
<tr>
<th>state</th>
<th>use</th>
<th>def</th>
<th>in</th>
<th>out</th>
<th>in</th>
<th>out</th>
<th>in</th>
<th>out</th>
<th>in</th>
<th>out</th>
<th>in</th>
<th>out</th>
<th>in</th>
<th>out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td></td>
<td>a</td>
<td>bc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td></td>
<td>b</td>
<td>a</td>
<td>ab</td>
<td>a</td>
<td>ab</td>
<td>a</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
</tr>
<tr>
<td>5</td>
<td>a</td>
<td></td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>a</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
<td>ab</td>
</tr>
<tr>
<td>6</td>
<td>c</td>
<td></td>
<td>a</td>
<td>a</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
<td>abc</td>
</tr>
<tr>
<td>7</td>
<td>c</td>
<td></td>
<td>c</td>
<td></td>
<td>c</td>
<td></td>
<td>c</td>
<td></td>
<td>c</td>
<td></td>
<td>c</td>
<td></td>
<td>c</td>
<td></td>
</tr>
</tbody>
</table>
Then we complete the diagram:

```plaintext
1: a = 0          Live: a, c
2: b = 0          Live: a, b, c
3: a = 10         Live: a, b, c
4: print b        Live: a, b
5: a = a + 1      Live: a, b
6: c = 2          Live: a, b, c
7: return c       Live: -
```
Question 5

As we see from the diagram above we can always have at most 3 live variables which we can store in the three registers without spilling.

The following coloring scheme simplifies the initial program to:

\[
\begin{align*}
r_1 &= 1 \\
r_3 &= 2 \\
r_2 &= r_1 + 1 \\
r_1 &= r_1 \\
r_2 &= r_2 + 1 \\
r_3 &= r_3 + r_2 \\
\text{return } r_1, r_3
\end{align*}
\]