CS 3721: Programming Languages Lab
Lab #9: Memory layout

Memory layout Instead of drawing a pictorial graph of the memory snapshot, the following describes each activation record in a textual format.

```
Example code:
1:    val x = 1;
2:    fun g(z) = x+z;
3:    fun h(z) =
4:               let x = 2 in
5:                  g(z);
6:    h(3);

Memory layout (runtime stack):
/* "->" means "points-to";  ":" means "has value" */
/* <global, code for g>  is the closure storage for function g*/
/* <h(3) : ?> means h(3) has an undefined value at the time*/
/* <global.h(3)> means the h(3) storage allocated in the AV of global*/
==================
  global : {
           Control Link -> NULL;
           Access Link -> NULL;
           Return Address -> OS code;
           Return Result Address ->  OS data;
           x  : 1 ;
           g  -> <global, code for g> ;
           h  -> <global, code for h> ;
           h(3) :  ? ;
          }
  h(3) : {
           Control Link -> global;
           Access Link -> global;
           Return Address -> <line after 6>;
           Return Result Address -> global.h(3);
           z  : 3;
          }

  h(3)-1: {
           Control Link -> h(3);
           Access Link -> h(3);
           x  : 2 ;
           g(z) : 4;
          }
  g(z) : {
           Control Link -> h(3);
```

```
 Access Link -> global;
 Return Address -> <line after 5>;
 Return Result Address -> h(3).g(z) ;
 z  : 3 ;
 x+z : 4;
}
```

1. Draw (in the textual format) the memory layout for the following ML code immediately before the first call to *mult* returns.

```
1:  let val x = 1; val y = 2
2:  in  let fun compose(f, g) =
3:                f(g(x));
4:              fun mult(x) =
5:                    x*y
6:      in  let val x = 3 in
7:          compose(mult, mult)
8:          end
9:      end
10: end;
```