UNIVERSITY OF LIVERPOOL

# COMP319 Dependency graph example

```
int salary=0;
float savingRate=0;
float pensionRate=0;

if (age<60)
{
   savingRate=0.08;
   pensionRate=0;
}
if (age>60) && (age<80) {
    pensionRate=0.1;
}
if (age>=80) {
   pensionRate=0.2;
}

float pension=salary*pensionRate;

savings=savings+salary*savingRate;
```

Labelling relevant nodes

```
(1) int salary=0;
(2) float savingRate=0;
(3) float pensionRate=0;

(4) if (age<60) {
(5)     savingRate=0.08;
(6)     pensionRate=0;

(7) if (age>60) && (age<80) {
(8)      pensionRate=0.1;

(9) if (age>=80) {
(10)     pensionRate=0.2;

(11) float pension=salary*pensionRate;

(12) savings=savings+salary*savingRate;
```

From entry to All nodes put control line (straight line)

From (12) there is a data dependency from (1), (2) and (5)

From (11) there is a data dependency from (1), (3), (6), (8) and (10)

From (4) to (5) and (6) two control lines (straight lines)
From (7) to (8) a control line (straight line)
From (9) to (10) a control line (straight line)

Now the def-order dependencies, these go from writes to the same variable which are controlled by the flow of execution

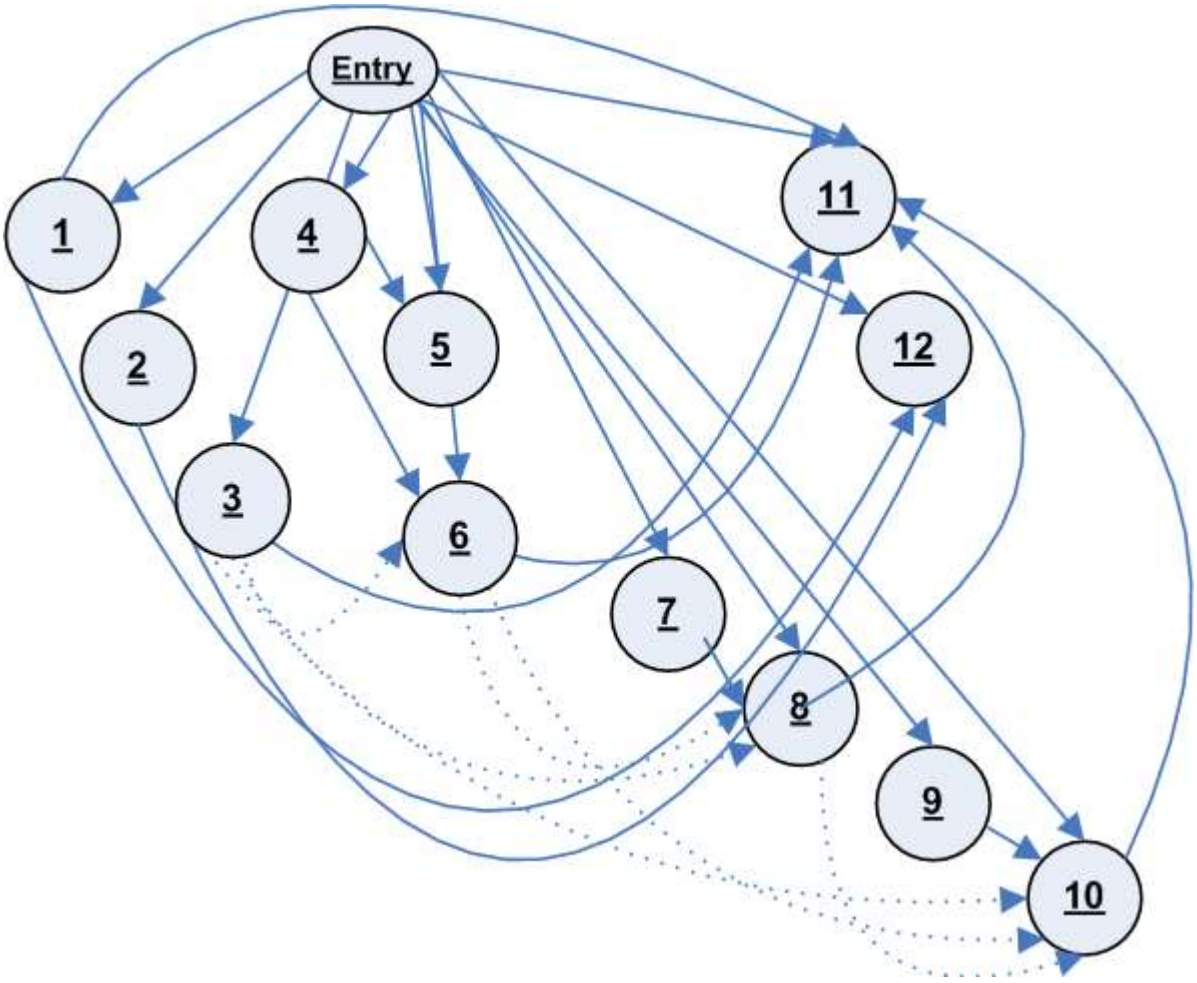
From (2) to (5) (one dotted line) As saving rate only used once

There are also def-orders for all statements that write to pension-rate
From (3) to (6) (one dotted line) As pension rate only used once
From (3) to (8) (one dotted line) As pension rate only used once
From (3) to (10) (one dotted line) As pension rate only used once

From (6) to (8) (one dotted line) As pension rate only used once
From (6) to (10) (one dotted line) As pension rate only used once
From (8) to (10) (one dotted line) As pension rate only used once

These def-orders define the dependencies of the variable that are controlled by the actual order of execution.

Solution (see solution over on the next page)


Entry
1
2
3
4
5
6
7
8
9
10
11
12


UNIVERSITY OF
LIVERPOOL