(a) Annotate the uses of a given variable with ‘D’ (definition), ‘R’ (reference) and ‘U’ (undefined—e.g. entering or leaving scope). Explain why three pairs of such uses may be regarded as anomalous data flow, giving brief example programs illustrating how each of these anomalies can represent a programmer error. Your answer should make use of the concept of path in a dataflow graph. [3 marks]

(b) Give a single program containing all three of the above anomalies and justify it as not containing any programmer error. [3 marks]

(c) By analogy with “on all paths” and “on some path” in dataflow analysis refine your definition of dataflow anomaly in part (a) into that of ‘must-anomaly’ and ‘may-anomaly’ indicating which, if either, of these corresponds to the previous definition. How many must-anomalies does your program in part (b) contain? [3 marks]

(d) Give two sets of dataflow equations, in the style used to define “live variables” or “available expressions”, which each calculate a set of variables for every program point. Explain how these can respectively be used to issue compiler warning messages of the form “variable ‘x’ may be read before being set” and “variable ‘x’ is definitely read before being set”. For both forms, state the initialisation of these sets of variables to be used when solving the dataflow equations. [7 marks]

(e) Now suppose the language supports indirect assignment to address-taken variables as in C. Explain briefly the refinements necessary to the above analyses. [2 marks]

(f) The wording (‘is definitely read’) of the second message in part (d) may be criticised when considering the program

```c
{ int x,y=0; while (...) y=f(y); print x; }
```

Comment briefly on this claim. [2 marks]