FORMATIVE EXERCISES (3)

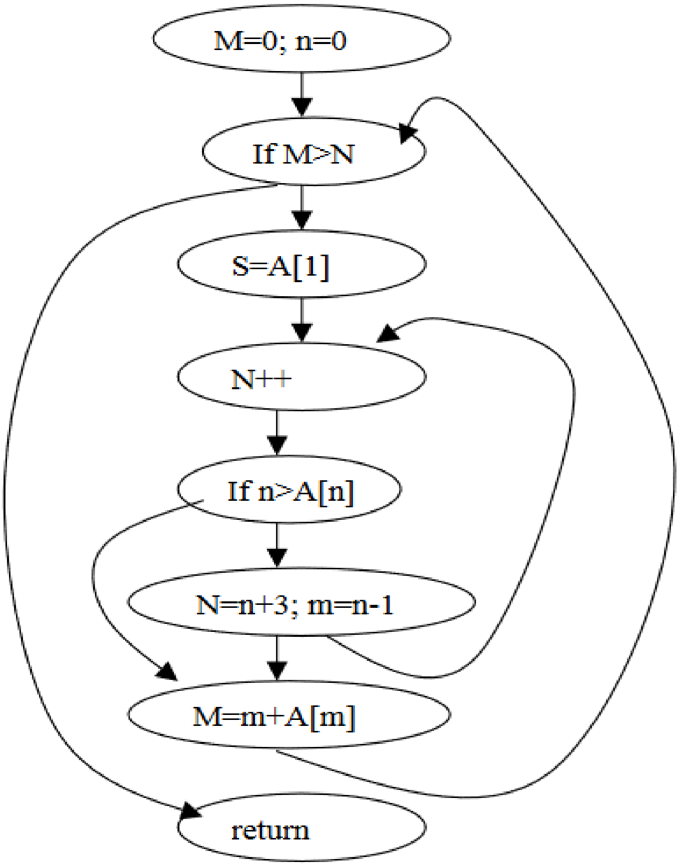
a) A possible solution (using only synthesized attributes) is the following:

G → E	G.val = E.val
E → E1 + T	E.val=E1.val + T.val
E → T	E.val=T.val
T → T1 * F	T.val=T1.val × F.val
T → F	T.val=F.val
F → (E)	F.val=E.val
F → digit	F.val=digit.lexval

The only assumptions are related to the lexical value of digit and the use of E1 and T1 (2nd and 4th rules) to specify the order of calculation of the attributes.

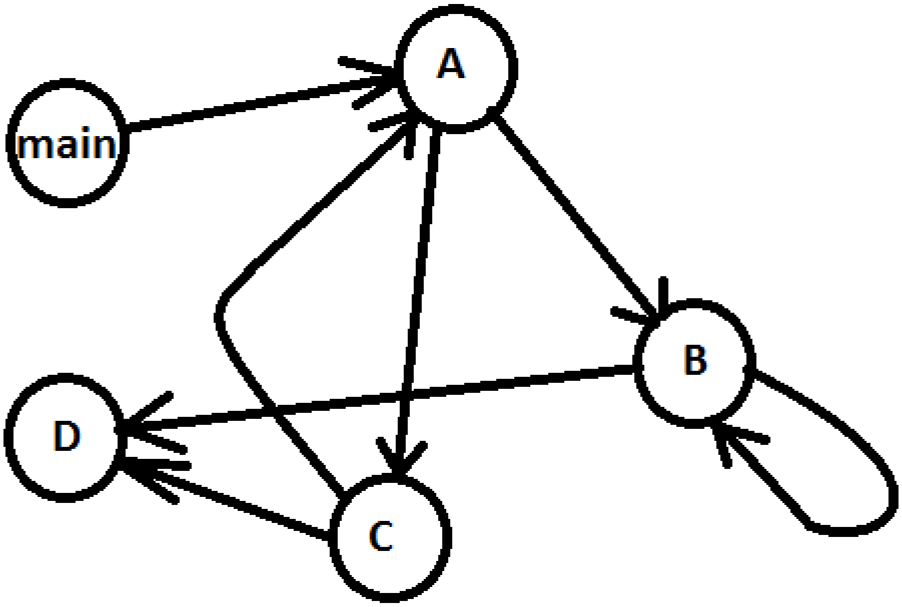
(5 marks)

b) See below. The important issues are that: (a) you group basic blocks in one node and (b) this is a directed graph (for completeness you should annotate edges following ifs with the specific condition that needs to hold to follow a specific edge.)



(5 marks)

c) First activation record (AR) is created for main(). Then, one AR for each of: A(-3), C(3), A(2), B(2), B(1), D(1), when the printf is reached. At this point, there are a total of 7 activation records in the stack. The call graph is a directed graph that should draw dependences between caller and callee functions. See below:



(5 marks)

d)
Since the hash function would take the remainder of the division by 2048, this means that only the last 11 bits of the 32-bit integer are used. Since we add zeroes at the end of 4-byte chunks, this means that all single-letter (and two-letter) variables would be mapped to the same position, that is position zero. This would create too many collisions. As a general rule, taking the remainder of a division by a power of 2 is really a bad choice for a hash function (and not only! ☺).
(5 marks)