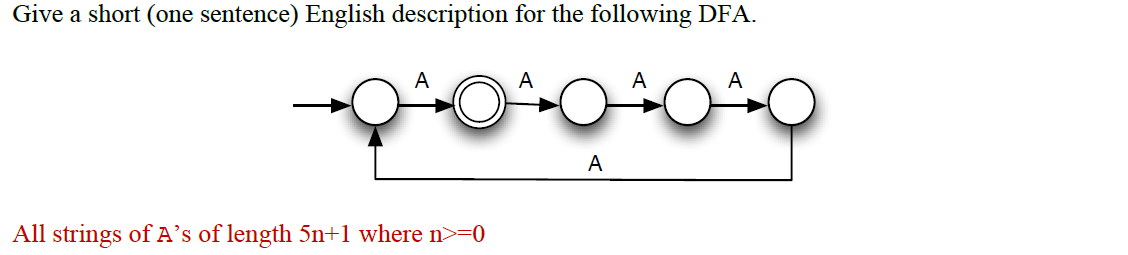
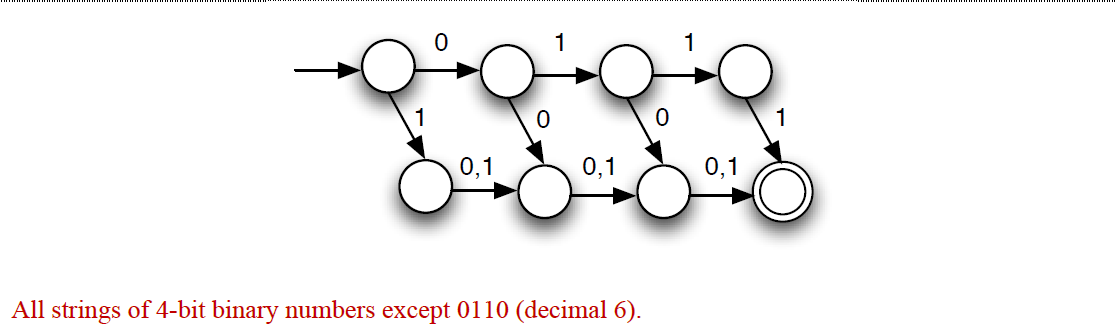
a m 1 (from tutorials)

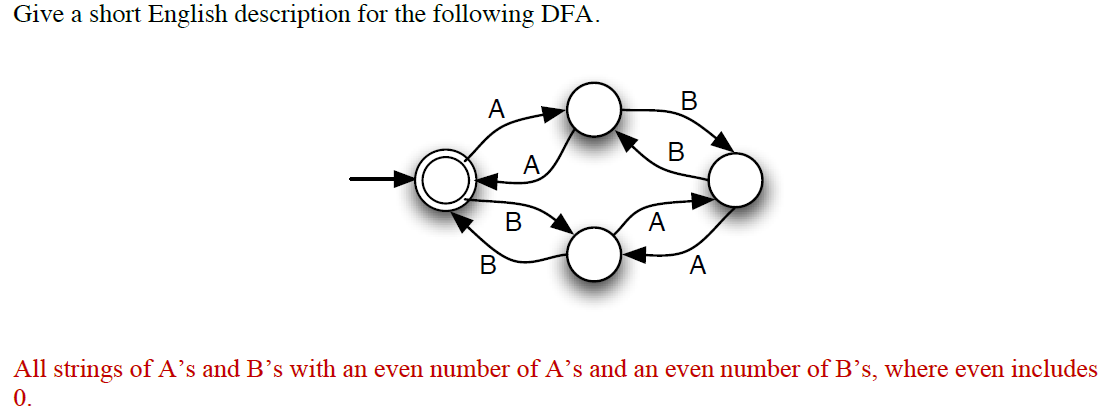
1a

i) 

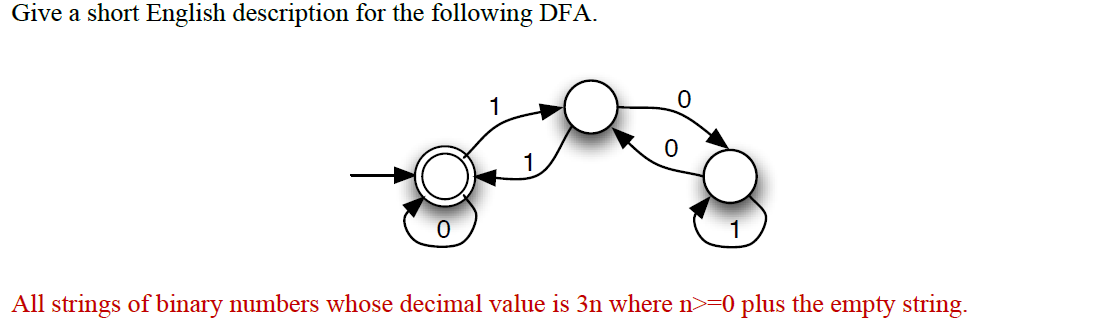
ii)



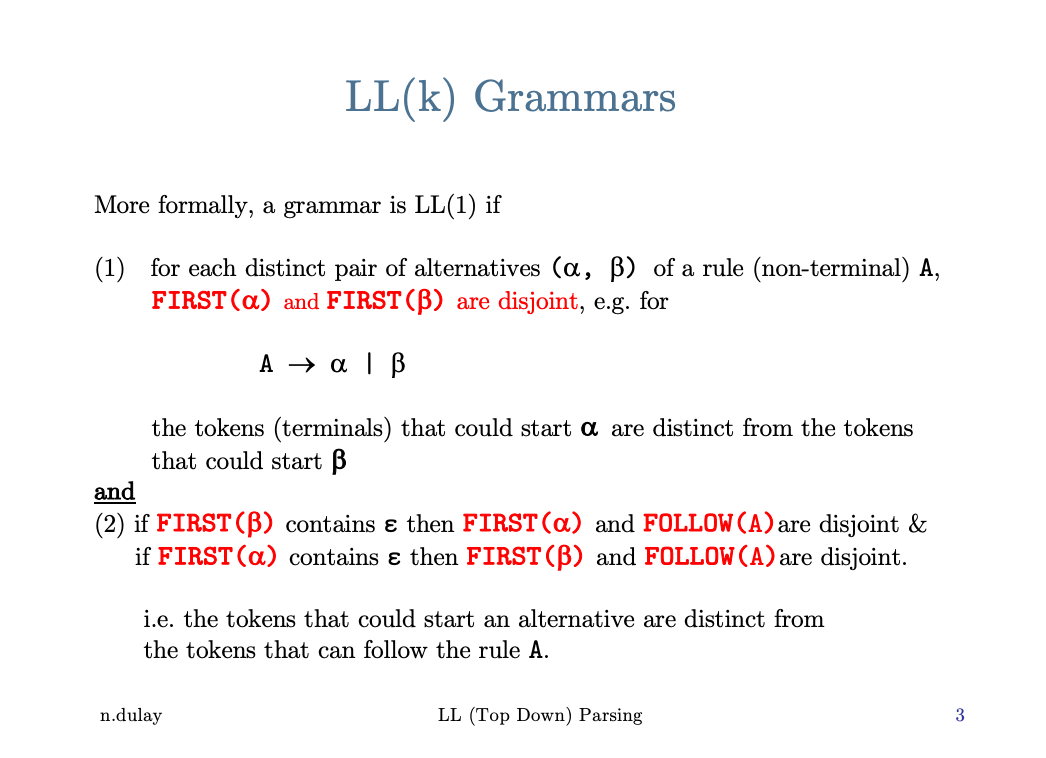
iii)



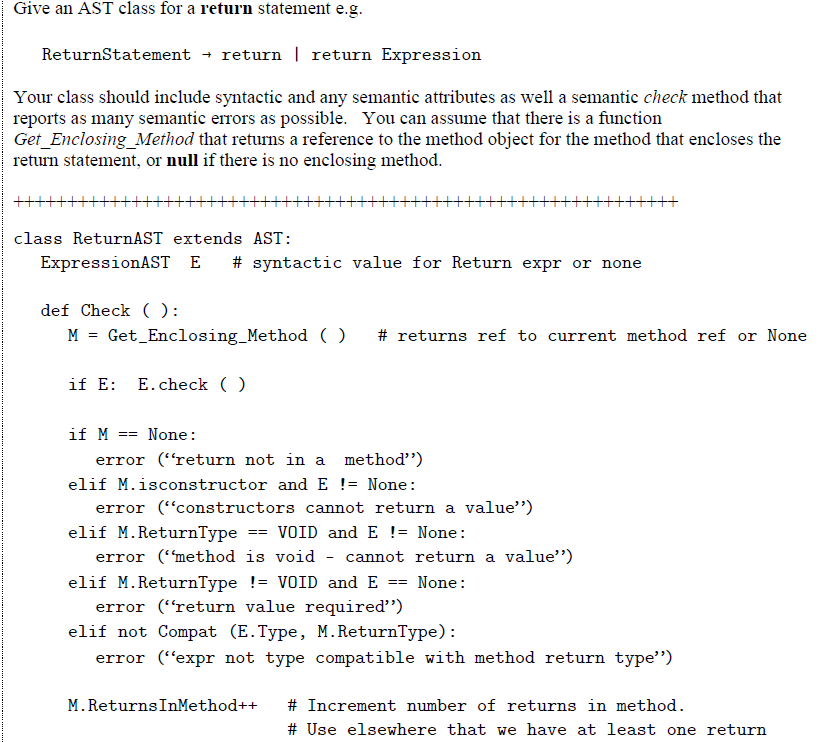
iv)



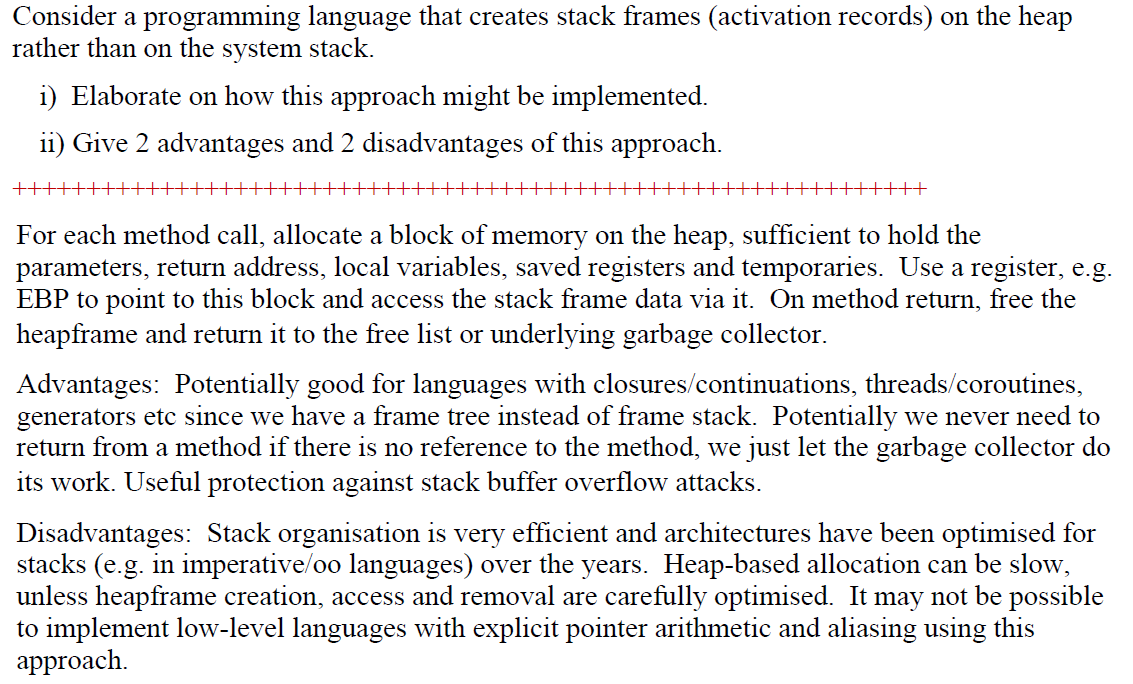
1b



1c



1d



2

1. translateStatement (While exp body)

= [Jump condLabel] ++

[Define bodyLabel] ++

translateStatement body ++

[Define condLabel]

translateExpression exp ++

[JTrue bodyLabel]

where

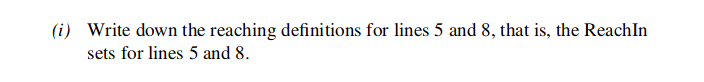
condLabel = label not used elsewhere

bodyLabel = another label not used elsewhere

b)

i)

Note this question has a mistake:



ReachIn(5) = {0,2,3,4,5,6,7,8 }

ReachIn(8) = {0,2,3,4,5,6,7,8}

ii)

s connected to i

i connected to s, s1, s2, s3, s4

s1, s2, s3, s4 all connected to each other and i, but not to s

iii) *Use graph colouring on the interference graph to only use 5 registers*

*Re-write code using R0 → R4 and s has same register as one of s1 → s4*

don’t declare s1, s2, s3, s4, s (do your own manual constant folding).

lines inside loop become:

B[i] = A[i][1] + A[i][2] + A[i][3] + A[i][4]

iv) Not sure what is actually loop invariant here? (someone help) I didn’t think anything was actually loop invariant since A and B are both indexed by i and i changes after each iteration.

v) If something was loop invariant, it would be lifted up out of the loop, most likely directly above it into an empty CFG node, and lie just below the CFG nodes for the declarations on line 1.

I didn’t think it was valid in this example. Couldn’t tell ya why though.

c)

Check function is not recursive or compiler will inline forever.

You would want to check that the local variables declared in the function don’t conflict with those in the scope you’re injecting the code into.

Bit of a shot in the dark but you might want to check that the functions return type is correct for where it is being called. e.g. if the function is void then it’s not being assigned to anything and if it has a return type the variable the return value is saved in matches the return type.

*Check that the function doesn’t use variables which are not in scope where you are injecting into (e.g. if function in another class with global variables which are not visible from line where it is called)*