1 LR(1)

Consider the following grammar, where \{a, b, c\} are terminal symbols:

\[
\begin{align*}
S & \rightarrow aXab & (1) \\
& \mid Y & (2) \\
X & \rightarrow bYa & (3) \\
& \mid \epsilon & (4) \\
Y & \rightarrow Sc & (5)
\end{align*}
\]

1. Construct the full LR(1) DFA, showing all items in each state.

2. Construct the LR(1) parsing table using the DFA. For the reduce actions, please use the provided enumeration of the productions in the grammar.

3. Show all steps required to parse the following string: \texttt{abaabccaab}
2 Activation records and scoping

In the following listing we show an excerpt of a program in a language that allows nested definition of functions, is *statically scoped* and uses *calls-by-value*:

```c
int foo(int a) {
    return 2 * a
}

int baz(int d, int e, int b) {
    int f, g;
    f = 2 * e + b;
    g = foo(f + d);
    return g;
}

int bar(int b, int c) {
    int h;
    h = baz(2 * b, c, d);
    return h + 42
}
```

Suppose we have in our `main()` function a call to `bar(1, 2)`. Assume that the compiler allocates all the variables (named and temporary) on the stack.

1. What will be the return value of this call?
2. Give a diagram of the state of the stack right before the return of the call to function `foo` on line 10. Your diagram should contain at least the following information regarding the calls to `foo`, `bar` and `baz` functions (including the initial call):
   (a) boundaries of the activation records
   (b) location of the actual parameters for the calls to these functions
   (c) location of the return values
   (d) location of the return addresses
   (e) location and contents of the control links
   (f) *Optional:* Location of temporary variables
   (g) *Optional:* Contents of actual parameters
3. Show where the values of the following variables are located in the stack and describe the code that should be generated by the compiler to access these values in the specified locations (including any points of reference, like the stack pointer). You may use as many registers as you like to calculate any addresses that are needed:
   (a) Variable `f` on line 8
   (b) Variable `e` on line 7