Exercise 6

Name: Caolan Barry Student Id: 15321767

For this exercise, I first decided how I wanted my Switch statement to work. What I always found awkward about Switch statements (The ones I have used anyways), is how you must break from them manually otherwise they would “fall through” and continue executing other cases until it “broke”. For my switch statement, I want it to break automatically by default instead of proceeding to execute other cases. If I want it to proceed to the next “case” the use of the command “continue;” will permit this.

The general usage for my switch statement is that it takes a simple expression and for the given resultant integer value it can switch to a single “case” that is defined in the switch statement, the case that it switches to is equivalent to the integer value equated. If there is no switch statement that pertains to this integer value the switch statement will switch to a default case that catches any integer value that has no case related to it.

For example: switch (2\*3+4) { … this instance of switch will attempt to switch to case 10 but will also switch to default if no case 10 exists.

I deduced that I would need to use a simple expression for the switch statement by looking at the attributed translation grammar.

At first I thought I could use a normal expression but upon examining the production rules I realised an expression takes the form:

Expr = SimExpr [RelOp SimExpr [ConditionalExpression] ]

An expression has the possibility of equating to a boolean value by the option of the RelOp

I then began to construct the production rules for my switch statements.

My production rules