Kardelen Demiral 2018400027
Berfin Şimşek 2018400009

DOCUMENTATION

The problem was basically generating a translator to translate our language to the LLVM IR code. Since we are unfamiliar with LLVM IR, it took some time to understand what it does at first. After we got the idea, we decided to divide our code into parts like allocation, assignment statements.

First of all, we print all the necessary lines like defining the main then we check all lines whether there is an error or not. There are 3 functions for error checking; "errorCatch", "errorCatchForExpressions" and "IsValidVariableName". First one checks whether while, if and print statements are written properly. Second one checks the expressions and the last one checks the variable names. While checking, we also push our variables to a vector in order to allocate after checking. If there is an error, we print the line that has the error and finish the execution. If not, we allocate the variables that we pushed to a vector.

After that, we get the lines one by one again in a while loop and decide its type like while or assignment etc. and print the necessary lines for each of them (br label..). If it is while, if or print we took the string inside the paranthesis, if it is an assignment statement we took the right part of the "=" as a string and send to a function called "muko". This function does all necessary things for all possible expressions. First, it calls the "infixToPostFix" function which separates every element of the expression and push them into a stack in the correct order of precedence. If stack has only one element and it is not choose function, it returns the value of the element or the temporary value that it is loaded. If it is choose, it calls the "choose" function that makes the program decide which expression should be returned according to the problem and returned the value of it. We used llvm "select" operator here. If stack has more than one element, it means there is an aritmetic operation. So it takes the elements from the stack in twos and calls the "operation" function which basically prints the correct statements for arithmetic operations. All necessary lines that should print for each statement has done in the functions so far. After "muko" function returns, we print the other necessary lines according to the statement types. When this big while loop finishes, we print the return statement and this is the end of the code.

We had troubles when we were writing the choose function. First we tried to write like how we write the if statements but it gave error because we were loading a temporary variable to another temporary variable. Then we improved our program with llvm select operation which makes our lives easier and it solved the error. Another problem occurred in the allocation part. We were allocating the variables that at the left part of the assignment at the beginning then when we see a variable that is not allocated we were allocating at that point. But we noticed that allocation in the while statements gives error so we changed our code to make all necessary allocation in another while at the beginning if there are not any error.

As an up of our code, we considered every possible error in a line so we have very detailed error checking functions. On the other hand, our code is repetitive. For example, we have a function who deletes the spaces in a line and we used this function basically everywhere even when it is unnecessary. Also our code is quite long because of the repetitions. For instance, we used almost exactly the same code in our "choose" function when we divide choose function into parts according to the commas in the error checking function also.

If we had more time, we could write a more efficient program but since the time is limited we focused on just implementing the code properly. Additionally, we might improve our code to be shorter and less repetitive.