HASAN BINGOLBALI 2019400276
OSMAN FEHMI ALBAYRAK 2018400198

We were asked to create a basic programming language. We were given a txt file and we were supposed to create Intermediate LLVM code accordingly. We should implement a function namely choose which takes 4 parameters and return an integer as well as being able to process four operations. We initially started by detecting errors. We divide the assignment operation as <var> = <expr> the while operation as while(<var>) etc.. Then, we write a function to check the validity of expr with the help of infix to postfix notation. Whenever we detect some unusual behavior in infix to postfix, we found out that there's something wrong in expression. After that, since, all the variables are globally accessible we used a while loop in order to find all the variables and allocate them memory at the beginning of an our LLVM code. We checked all the assignment operations, if , while and print conditions and the choose function. We implemented checking choose function recursively. We extracted the 4 expressions and evaluate them again so that we have been able to deal with nested chooses and the operations inside the choose function. When it comes to printing LLVM code , we implemented infix to postfix notation for the expressions and then manipulate the infixToPostfix when we encounter an operator. And in this postfix notation Choose was thought to be an operator which takes 4 previous inputs. For example;

choose(a,b,c,d) was represented as ————> a b c d choose

And after the processing choose it was turned into —> temp1

When there is only one element left in the postfix we were assigning it to a variable or checking whether it is 0 depending on the context. We implemented choose in condition, body , end blocks. We also defined and choose number so that our llvm code wouldn't label to previous condition of the choose. We implemented if and while with the same approach. We also change the our %temp name because it could collide with the name of a variable. Therefore, we renamed it ha_o which is the composition of our initials (Hasan and Osman).

When it comes to what we would focus on more if we had time, we made a lots of copy-paste and because of that our code is as long as thousand lines. We could have used more functional structures so

that our code would be shorter; hence, more readable and more elegant.