Compiler Construction

BPDC

Lab Assignment (Weightage 05)

Deadline: 30/04/2020

1 A Language Translator from $\mathcal{C}$ to C

In this assignment, the task is to design a translator that would take a program written in the language $\mathcal{C}$ (some $\mathcal{C}$ustomized language) as input and outputs an equivalent C program. The constructs defining the language $\mathcal{C}$ together with some examples are given below. Our job is to convert each of those constructs to an equivalent C construct.
We could assume the language $\mathcal{C}$ not to support function calls and libraries and hence your C program would always follow the template:

```
#include<stdio.h>
int main()
{
    statements;
}
```

2 Constructs in $\mathcal{C}$ language

2.1 The symbol ; separates statements as in C

2.2 Comments

Block Comments are usually enclosed in **(*** and ***)** (C equivalent for /* comments */).

2.3 Declaration statements

1. **Var var_name$_1$,var_name$_2$,...,var_name$_n$:type;** where Var is keyword indicating these are variables with names var_name_i, each being of data type *type*, which can be $int, char, float$ or *double*.
2. **Var var_name:Array[SIZE$_1$][SIZE$_2$] Of type;** declares a 2D array where Var, Array and Of are keywords and *type*, as usual, can be $int, char, float$ or *double*.

The language doesn't support initialization during declaration.

2.4 Operators:

1. := is the assignment operator ($a = 10$; in C would be $a := 10$;).
2. **And** is equivalent to && in C.
3. **Or** is equivalent to || in C.
4. **Not** is equivalent to !(logical operator) in C.
5. Everything else is the same as in C.

2.5 Conditional statements:

1. **If** (*condition*) **Then** *statements* **EndIf**
2. **If** (*condition*) **Then** *statements* **Else** *statements* **EndIf**
3. **If** (*condition*) **Then** *statements* **ElseIf** (*condition*) ... **Else** *statements* **EndIf** //nested if, need not end in an *Else* block.

In addition, our $\mathcal{C}$ supports **goto** operator (same as the one in C) subsuming the actions of both *break* and *continue* operators in C. Correspondingly, you could have labels in your program (LABEL followed by ':' operator, refer goto operator in C for better understanding).

2.6 Loop structures:

1. **While** (*condition*) **Do** *statements* **EndWhile**
2. **Repeat** *statements* **Until**(*condition*) //Corresponds to dowhile(*condition*) in C
3. **For**($i \leftarrow 1$ **To** $LIMIT$) **Do** *statements* **EndFor** //By default, step count is 1. O

2.7 Input and Output:

1. **Read**($var_1, var_2, \ldots, var_n$)**;** //In contrast to C, the function Read() doesn't bother about verifying the data types associated with variables.
2. **Write**($var_1, var_2, \ldots, var_n$)**;** //Instead of variables, we could even have interactive messages as static strings enclosed in single quotes (say, 'The value of x is') as arguments to function Write

2.8 The main procedure:

Procedure Main()//
Begin:
$statement_1$
$statement_2$
.
.
.
$statement_n$
End:

3 Sample Translation

3.1 $\mathcal{C}$ Program

```
Procedure Main()
Begin:
        var num:int;
        var i,flag:int;
        flag:=0;
        Write('Enter the value to be checked:');
        Read(num);
        For(i<-2 To num-1) Do
           If(num%i==0)
```

```
            flag:=1;
            goto Exit;
          EndIf
        EndFor
Exit:
        If(flag==1)
            Write('The value ',num,' is not prime');
        Else
            Write('The value ',num,' is prime');
        EndIf
End:
```

3.2 Equivalent *C* Program

```
#include<stdio.h>
int main()
{
        int num;
        int i,flag;
        flag=0;
        printf("Enter the value to be checked:");
        scanf("%d",&num);
        For(i=2; i<= num-1;i++) {
          if(num%i==0){
            flag=1;
            goto Exit;
          }
        }
Exit:
        if(flag==1){
            printf("The value %d is not prime");
        }else{
            printf("The value %d is prime");
        }
}
```