CSC 365: Tutorial #1

Anup Adhikari
tutor@anup.pro.np

Compiler Design and Construction July 20, 2022

1 Introduction to Compiler Design

This tutorial section is focussed to know about the basics of Compiler blocks.

Question 1

Compilers are broken into several chunks called passes that communicate with one another via temporary files. Justify the construction of compiler as a complex program.

Question 2

Define Lexer. Describe the role of lexical analysis in compiler construction.

Question 3

Define parsing. Compare the parsing process of English sentence: 'Ram sees Laxman flee' with the parsing process of Expression (A * B - C * D).

Question 4

What is Code Generation in compilation process? Discuss on the basic principles.

Question 5

Write short notes on:

1. Context Free Grammar
2. Backus-Naur Form
3. Syntax Diagram

Question 6

Write a grammar that recognizes a C variable declaration made up of the following keywords:

```
int  char long float double signed unsigned short const volatile
```

and a variable name.

Question 7

Write a grammar that recognizes a C variable declaration made up only of legal combinations of the following keywords:

```
int char long float double signed unsigned short const volatile
```

and a variable name. The grammar should be able to accept all such legal declarations. For example, all the following should be accepted:

```
volatile unsigned long int x;
unsigned long volatile int x;
long unsigned volatile int x;
long volatile unsigned int x;
```

but something like this should not be accepted:

```
unsigned signed short long x;
```

Question 8

Write a grammar (and a recursive-descent compiler for that grammar) that translates an English description of a C variable into a C-style variable declaration. For example, the input:

```
x is a pointer to an array of 10 pointers to functions that return int.
y is an array of 10 floats.
z is a pointer to a struct of type a struct.
```

should be translated to:

```
int (* (*x) [10]) ();
float y[10];
struct a_struct *z;
```