ANDERSON BAIDEN RICHARD (8122612)

CSM 484 (INTRO TO COMPILERS)

Question 1.



The following describes in detail, the phases of the compilation process as shown in the diagram above:

**Lexical Analysis**

The first phase of scanner works as a text scanner. This phase scans the source code written by the programmer as a stream of characters and converts it into meaningful lexemes. Lexical analyzer represents these lexemes in the form of tokens as: token-name, attribute-value.

**Syntax Analysis**

The next phase is called the syntax analysis or *parsing*. It takes the token produced by lexical analysis as input and generates a parse tree (or syntax tree). In this phase, token arrangements are checked against the source code grammar, i.e. the parser checks if the expression made by the tokens is syntactically correct.

**Semantic Analysis**

Semantic analysis checks whether the parse tree constructed follows the rules of language. For example, assignment of values is between compatible data types, and adding string to an integer. Also, the semantic analyzer keeps track of identifiers, their types and expressions; whether identifiers are declared before use or not etc. The semantic analyzer produces an annotated syntax tree as an output.

**Intermediate Code Generation**

After semantic analysis the compiler generates an intermediate code of the source code for the target machine. It represents a program for some abstract machine. It is in between the high-level language and the machine language. This intermediate code should be generated in such a way that it makes it easier to be translated into the target machine code.

**Code Optimization**

The next phase does code optimization of the intermediate code. Optimization can be assumed as something that removes unnecessary code lines, and arranges the sequence of statements in order to speed up the program execution without wasting resources (CPU, memory).

**Code Generation**

In this phase, the code generator takes the optimized representation of the intermediate code and maps it to the target machine language. The code generator translates the intermediate code into a sequence of (generally) re-locatable machine code. Sequence of instructions of machine code performs the task as the intermediate code would do.

**Symbol Table**

It is a data-structure maintained throughout all the phases of a compiler. All the identifier's names along with their types are stored here. The symbol table makes it easier for the compiler to quickly search the identifier record and retrieve it. The symbol table is also used for scope management.

Question 2.  
A translator is a computer program which translates a coded program in a particular language into an equivalent functioning program in a different programming language without losing its logical and functional structure.

Examples of translators are:  
Source-to-source compilers, Boomerang decompilers and interpreters.

Question 3.  
**A linker** is a specific computer program whose function is to combine object modules to form an executable program.

**A loader** is a piece of an operating system whose function is to load programs and libraries.

**A preprocessor** is a computer program which simply converts input data into output which will be used by another program as an input data.

**An assembly language** is a low-level programming language for computers consisting of a strong correspondence between the language itself and the computer architecture’s machine code instructions.

**Machine language** is a low-level programming language understood by computers. It is actually the lowest level.

Question 4.  
**Tokens** are sequence of characters which can be a classified as single logical entities in syntax of programming languages. E.g. Keywords, Identifiers, Operators.

**Lexicons** are a group of words used to create a programming language.

A **lexeme** is a sequence of characters in the source program of a computer that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.