**Assignment 4**

**1、What sequence of numbers would be printed if the procedure named xxx as described below were executed with the value of N being 2?**

**Answer：**

The sequence of numbers, 2 4 5 4 2, will be printed.

**2、If numeric values are represented in two’s complement notation, does the following program represent an infinite process? Explain your answer.**

**Answer：**

The provided program does not represent an infinite process. Initially, X is set to the value 2. In two's complement notation, this is a positive value. The loop condition checks whether X is greater than 0. As long as X is positive, the loop will continue.

In each iteration of the loop, X is incremented by 1 (X ← X + 1). Since X started as a positive value 2, the addition will continue to make it more positive. However, there is a limit to how large a positive number can be represented in a fixed number of bits.

Eventually, X will reach the maximum positive value that can be represented in the given number of bits. At that point, the addition operation will overflow, resulting in a wraparound to the minimum representable value (due to two's complement arithmetic). The minimum representable positive value is 0. When X becomes 0, the loop condition (X > 0) becomes false, and the loop exits.

**3、The following program segment is designed to compute the product of two nonnegative integers X and Y by accumulating the sum of X copies of Y – that is, 3 times 4 is computed by accumulating the sum of three 4s. Is the program correct? Why?**

**Answer：**

The program is wrong. Because when x is 0 and y is not 0, the program execution result is incorrect. The corrected version of the program should be:

Product <- 0;

Count <- 1;

while (Count <= X) do

(Product <- Product + Y;

Count <- Count + 1)

**4、Explain the distinction between the imperative and declarative programming paradigms.**

**Answer：**

The imperative and declarative programming paradigms are two fundamentally different approaches to writing and organizing code. Here's an explanation of the key distinctions between them:

Imperative Programming:

How to Achieve a Result: In imperative programming, the focus is on describing a sequence of steps to achieve a specific result. The programmer provides detailed instructions on how the computer should perform a task.

State Modification: Imperative programs often involve changing the state of variables and the program itself. The emphasis is on how to perform computations and manipulate the program's state.

Declarative Programming:

What to Achieve: Declarative programming, on the other hand, focuses on describing the result or the desired outcome without explicitly specifying the steps to reach that outcome. It emphasizes what needs to be achieved rather than how to achieve it.

Statelessness: Declarative programs tend to be more stateless. Instead of changing the state of variables, they express relationships and constraints between different parts of the program.

Key Distinctions:

Readability and Maintainability: Declarative code is often considered more readable and maintainable because it abstracts away low-level implementation details. It allows the programmer to focus on the "what" rather than the "how."

Parallelism and Optimization: Declarative code can be more amenable to parallelism and optimization because the compiler or interpreter has more flexibility in rearranging and optimizing the code.

Expressiveness: Declarative programming can be more expressive for certain types of problems, such as data transformations and queries, where expressing the desired outcome is more important than detailing the steps.

In practice, many programming languages and paradigms incorporate elements of both imperative and declarative styles, and the choice often depends on the nature of the problem being solved.

**5、Explain the distinction between a procedure and a function.**

**Answer：**

Procedure:

A procedure is a set of instructions that performs a specific task or action.It is a subroutine or a routine that may take some inputs (parameters), execute a series of statements, and perform actions, but it does not return a value.Procedures are typically used for tasks that involve performing actions or operations without necessarily producing a result that needs to be used elsewhere in the program.In some programming languages, procedures are also referred to as void functions.

Function:A function is also a block of code that performs a specific task or action, but it differs from a procedure in that it returns a value.Functions take input parameters, process them, and then produce a result that can be used elsewhere in the program.The return value is often used in expressions or assigned to variables.Functions are used when there is a need to compute and return a value.

In summary, the primary distinction lies in whether or not the block of code produces a return value. Procedures are used for tasks that involve actions without a direct need for a return value, while functions are used when there is a need to compute and return a result. The specific terminology and syntax may vary across programming languages, but the conceptual difference remains consistent.

**6、Briefly describe the task of each of the following.**

**A. Lexical analyzer**

**B. Parser**

**C. Code Generator**

**Answer：**

A. Lexical Analyzer:

The lexical analyzer, also known as a lexer or tokenizer, is responsible for breaking the source code into tokens. Tokens are the smallest units of a programming language, such as keywords, identifiers, literals, and operators. The lexical analyzer removes comments and whitespace, producing a stream of tokens that is easier for the parser to process.

B. Parser:

The parser takes the stream of tokens generated by the lexical analyzer and builds a hierarchical structure called a parse tree or abstract syntax tree (AST). It checks whether the arrangement of tokens conforms to the grammatical rules of the programming language. The parser essentially determines the syntactic structure of the code, ensuring it follows the specified syntax rules.

C. Code Generator:

The code generator translates the parse tree or AST produced by the parser into an intermediate code or machine code. It maps the high-level, human-readable code into a form that can be executed by the computer's hardware. This involves optimizing the code for efficiency and generating instructions that represent the intended behavior of the program. The code generator is a crucial component in the compilation process, bridging the gap between the abstract representation of the code and its executable form.

**7、Suppose the procedure Modify is defined as: procedure Modify (Y) Also suppose that X is a global variable. If parameters are passed by value, what will be printed when the following program segment is executed? What if parameters are passed by reference?**

**Answer：**

If parameters are passed by value, the output will be:

Value of X: 5

Value of Y: 9

If parameters are passed by reference, the output will be:

Value of X: 9

Value of Y: 9

In summary, if parameters are passed by value, the global variable X remains unchanged outside the procedure. If parameters are passed by reference, changes made to Y inside the procedure also affect the global variable X.

**8、The insertion sort algorithm is an example of an algorithm in which of the following classes? And Why?**

**• A. O(lg n) B. Θ(n2)**

**• C. O(n lg n) D. O( n2 )**

**Answer：**

The insertion sort algorithm belongs to the class represented by B. Θ(n^2).

Explanation:

Insertion sort is a comparison-based sorting algorithm with a time complexity of Θ(n^2) in the worst and average cases. The algorithm iterates through the list, repeatedly taking one element and inserting it into the correct position relative to the previously sorted part of the list. The worst-case time complexity occurs when the input list is in reverse order, causing maximum comparisons and swaps.

Options:

A. O(lg n): This is the time complexity of algorithms like binary search. Insertion sort is not in this class.

B. Θ(n^2): This accurately represents the time complexity of insertion sort.

C. O(n lg n): This is the time complexity of more efficient algorithms like merge sort and heap sort, not insertion sort.

D. O(n^2): This is the same as Θ(n^2) and is the correct class for insertion sort.

Therefore, the correct choice is B. Θ(n^2).