**Instructions**: You must read the material and create an outline of the topics in your OWN words.  Do not copy the text from the tutorials into your notes. Make sure your outline contains notes for each subsection of the reading assignment. Thoroughly cover each topic to show you have a firm understanding of the programming concept or construct.

| **Ques** | **NOTES:** |
| --- | --- |
| What is Pseudocode?  Key Points to Note  Pseudocode vs Actual Code  Why Use Pseudocode?  How to write Pseudocode?  Standard Conventions to Write Pseudocode  Pseudocode for Different Statements  Special Keywords  Conditional Statements  FizzBuzz Pseudocode  Advantages and Disadvantages of Pseusocode  How to Write Pseudocode as a Beginner?  10 Common Pseudocode Examples Every Programmer Must Try | This is a simple way of representing complex algorithms making it easy for anyone to read and understand the algorithm.  It uses simple English words, expressions, and simple programming constructs to represent the parts of any algorithm and although there is no set syntax rule certain simple conventions are used to make it universally appealing. It is seen a first step towards writing a code to solve a problem.  Flowcharts, drakon-charts and Unified Modelling Language (UML) charts are considered graphical alternatives to pseudocode with the disadvantage of taking up more space on paper.   * It is not bound to any programming language. * It is used as a learning and reasoning tool by programmers and developers to help them write good code. * It can be compiled or executed by any assembler, compiler and interpreter. * It does not follow any strict structure and syntax.   Using a simple example of some code to check if a user entered an even or odd number we can clearly see the difference between pseudocode and actual code:  C++ Programming  Int main()  {  int num;  cout<<”Enter a number”;  cin>>num;  if(num % 2 == 0)  cout<<”Even Number”;  else  cout<<”Odd Number”;  }  Pseudocode  BGEIN  Num = INPUT: “Enter a number”  IF num MOD 2 == 0  print “Even Number”  ELSE  print “Odd Number”  END  When embarking on any project it is often required to have a blueprint created that would make the project easier to understand and hence execute. Here are some reasons why pseudocode is used:   * It is easy to read, so it makes it easy for programmers to share ideas with non-technical folks. * It simplifies code construction as all the parts are clearly seen and quickly implemented by any with basic programming knowledge. * It is a middle ground between an algorithm and the actual code. * It can help find errors in the algorithm and make corrections, resulting in saved time.  1. **Use a plain text editor**: any plain text editor is good enough to write pseudocode. 2. **Define the Purpose or Goal of the Process**: a problem statement has to be defined to help you and others understand the purpose of the pseudocode. 3. **Logically outline the Steps:** all the steps should be sequentially outlined, putting one action per line. This will make it easy for the end user to translate it. 4. **White Space and Indentation:** Thisshould be used to make your pseudocode very readable and clean. 5. **Commands Should be Capitalized:** keywords that will be the same in the actual code should be capitalized. These include works like “IF,” “ELSE,” etc. 6. **Use Simple Terms:** the essence of pseudocode is to make it easy to understand a process; hence, simple terms should be used when writing it. 7. **Complete Description of The Process:** writing the steps can seem very verbose at times but it is best to describe every aspect of the process and avoid using shortcuts that can create confusion for less experienced programmers. For example, rather than “**IF n/2==0”, you should write “IF a number is completely divisible by 2**.” 8. **Organize Using Sections:** curly brackets should be used to create sections in our pseudocode, and “//” should be used to create comments where needed. 9. **Cross Verify:** make sure you go through your work to ensure all the parts of the process are covered and it is easy to read and understand.   Pseudocode doesn’t have a strict syntax rule, but certain conventions are generally followed. These include:   * Always use capital letters for keywords such as “IF,” “ELSE,” etc. * Write one statement per line. * Use indentation for the body of the pseudocode; this helps show the relationship between each line. * Always use plain English or your local language to describe what you are trying to do.   Operators:   1. Assignment Operator: =,<- or := 2. Comparison Operator: ==, !=, <, >, <=, and >= 3. Arithmetic Operator: +,-,\*,/,MOD(%) 4. Logical Operator: AND, NOT, OR 5. Sum, Product: ?? 6. START: Starts the pseudocode. 7. INPUT: To take input from the user. 8. PRINT: Print something to the screen or any other output device. 9. READ/GET: To get information from a file. 10. SET/INIT: Initialize a value 11. INCREMENT, BUMP: To increase the value of a variable earlier created, equivalent to a++, =+, depending on you programming language. 12. DECREMENT: To decrease the value of a variable earlier created, equivalent to a--, =-, depending on your programming language. 13. COMPUTE, CALCULATE, DETERMINE: To calculate the result of the expression. 14. ***IF:***   IF condition  THEN if body  ENDIF   1. ***IF..ELSE:***   IF condition THEN  If body  ELSE  else body  ENDIF   1. ***IF…ELSE IF…ELSE:***   IF condition statement THEN  If body  ELSE IF condition THEN  else if statement  ELSE  else body  ENDIF  Example1:  age = INPUT : “Enter your age”  IF age is greater then 18  PRINT “adult”  ELSE  PRINT “Under age”  Example2:  age = INPUT : "Enter Your age"  IF age is equal to 18 THEN  PRINT "under check"  ELSE IF age is greater than 18  PRINT "Give entry"  ELSE  PRINT "under age"  ENDIF  ***Iterators:***   1. ***For loops:***   FOR initial\_value TO end\_value  for body  ENDFOR  Example:  FOR i -> 0 to 20  PRINT i  ENDFOR   1. ***While loop:***   WHILE condition  while body  ENDWHILE  Example:  i : 0  WHILE i <= 20  PRINT i  ENDWHILE  ***Functions:***    FUNTION function\_name(parameters)  function body  RETURN value  ENDFUNCTION  Example:  FUNCTION add( para1, para2 )  result: para1+para2  RETURN result  ENDFUNCTION add  Here is an example of pseudocode and it’s translation to actual code:  Pseudocode:  BEGIN  num = 1  FOR num -> 1 to 20  IF num MOD 15 == 0 THEN  PRINT "FizzBuzz"  ELSE IF num MOD 3 == 0  PRINT "Fizz"  ELSE IF num MOD 5 == 0  PRINT "Buzz"  ELSE  PRINT num  ENDIF  ENDFOR  END  Actual Code:  for num in range(1, 21):  if (num % 15) == 0:  print("FizzBuzz")  elif num % 3 == 0:  print("Fizz")  elif num % 5 == 0:  print("Buzz")  else:  print(num)  Advantages:   * It increases code readability. * It makes it easy to write algorithms. * It focuses on the program logic instead of any specific syntax. * It explains every line of what must be done.   Disadvantages:   * It doesn’t use any formal format, which makes part of it confusing as each programmer uses its own code. * Lacks a visual representation of the logic, making it somewhat difficult for novices to understand. * Extra documentation is still required as it may not be able to explain everything.   As a beginner, it can be scary to learn to use pseudocode, but it should be noted that it’s not a requirement. It is highly recommended as it makes communicating your ideas to colleagues easier and coding easier.   1. Add Two Numbers   BEGIN  Declare variables num1, num2, and sum  num1 = INPUT: "Enter the first number"  num2 = INPUT: "Enter the second number"  sum = num1 + num2  OUTPUT sum  END   1. Calculate the Area and Perimeter of a Square   BEGIN  Declare variables length, area, and perimeter  length = INPUT: "Enter the side of a square"  area = length \* length  perimeter = 4 \* length  OUTPUT area  OUTPUT perimeter  END   1. Calculate the Area and Perimeter of a Circle   BEGIN  Declare variables radius, area, and perimeter  radius = INPUT: "Enter the radius of a circle"  area = 3.14 \* radius \* radius  perimeter = 2 \* 3.14 \* radius  OUTPUT area  OUTPUT perimeter  END   1. Calculate Sales Tax   BEGIN  Declare variables price, taxrate, tax, and total  price = INPUT: "Enter the product price"  taxrate = INPUT: "Enter the tax rate"  tax = price \* taxRate/100  total = price + tax  OUTPUT tax  OUTPUT total  END   1. Check if Your are Eligible for a Driving License   BEGIN  Declare a variable age  age = INPUT: "Enter your age"  IF age >= 16 THEN  PRINT "You are eligible for a driving license"  ELSE  PRINT "You are not eligible for a driving license"  ENDIF  END |
| Flowchart  Flowchart Overview  History of Flowchart  Types of Flowcharts  Building Blocks of Flowcharts  Parallel Processing in Flowcharts  Diagramming  Significance of Flowcharts | 1. Find the Largest of Three Numbers   BEGIN  Declare variables num1, num2, and num3  num1 = INPUT: "Enter the first number"  num2 = INPUT: "Enter the second number"  num3 = INPUT: "Enter the third number"  IF num1>num2 AND num1>num3 THEN  PRINT "num1 is the largest number"  ELSE IF num2>num3 THEN  PRINT "num2 is the largest number"  ELSE  PRINT "num3 is the largest number"  ENDIF  END   1. Print Number from 1 to 100   BEGIN  Declare a variable counter  FOR counter -> 1 to 100 STEP 1 DO  OUTPUT counter  ENDFOR  END   1. Read the First 50 Numbers and Find Their Sum   BEGIN  Declare variables counter  Initialize a variable sum = 0  FOR counter -> 1 to 50 STEP 1 DO  sum = sum + counter  ENDFOR  OUTPUT sum  END   1. Find the Sum of All Elements of an Array   BEGIN  Initialize variables i = 0, n = 5, and sum = 0  Initialize an ARRAY num = {4, 3, 6, 7, 9}  FOR i -> 0 to n-1 STEP 1 DO  sum = sum + num[i]  ENDFOR  OUTPUT sum  END   1. Swap Two Variables   BEGIN  Declare variables num1, num2, and temp  num1 = INPUT: "Enter the first number"  num2 = INPUT: "Enter the second number"  temp = num1  num1 = num2  num2 = temp  OUTPUT num1  OUTPUT num 2  END  This has been described as a diagram that represents a workflow or a process. It is used to represent the step-by-step approach to solving a problem. It uses boxes of various kinds to represent operations and sections of an algorithm, it does this in an orderly fashion with a general flow of top to bottom and left to right. They are used in various fields.  They are used to design, and document simple processes or programs and utilize the following main symbols:   * A process step, usually called an activity, and denoted by a rectangular box, and * A decision step denoted by a diamond.   When it is referred to as cross-functional, it is used to describe the control of different organizational units. They represent certain aspects of processes and are usually accompanied by other types of diagrams as pointed out by various researchers.  Several alternative names are used to reference it, such as flow chart, process flowchart, functional flowchart, process map, process chart, functional process chart, business process model, process model, process flow diagram, “work flow” diagram, and business flow diagram. It should be noted that the terms “flowchart” and “flow chart” are used interchangeably.  The first use of the term was recorded in 1921 by Frank and Lilian Gilbert. It was widely adopted by many and modified to suit their professions. It became a popular tool for describing computer algorithms, providing many advantages to programmers. Until the early 21st century, flowcharts were still used to describe computer algorithms and modern techniques such as UML activity diagrams and Drakon-charts were considered extensions of the flowchart.  Several authors have classified flowcharts based on various criteria such as groups (managers, system analysts, and clerks), which provide the following types. Here Stemeckert (2003) named the following types:   * *Document flowcharts*, showing controls over a document-flow through a system * *Data flowcharts*, showing controls over a data-flow in a system * *System flowcharts*, showing controls at a physical or resource level * *Program flowchart*, showing the controls in a program within a system   Others gave the following types of system flowcharts:   * Andrew Veronis (1978): system flowchart, general flowchart, and detailed flowchart. * Marilyn Bohl(1978): system flowcharts and program flowcharts. * Mark A. Fryman (2001): Decision flowcharts, logic flowcharts, systems flowcharts, product flowcharts, and process flowcharts.   Reversible Flowcharts:  These are flowcharts that are used to represent reversible computing processes.  Common symbols:  The American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) jointly produced a standard set of symbols for use in flowcharts with the latter formerly adopting that of the former in 1970.  Two horizontal lines represent this at the beginning or ending of simultaneous operations.  Most drawing programs can be used to create flowchart diagrams, but these types do not have any underlying databases to share data. Software packages exist that create flowcharts from program codes directly.  It helps to break down a complex problem into simple parts, making writing code much easier. Each step represents an independent implementation of the overall process and helps to give a better picture of the project at hand. They are often employed alongside other types of diagrams to help take care of it’s inability to represent a process fully. |

Summary:

I learned that pseudocode is very important in helping me understand a problem and write code to solve that problem, and it showed the various aspects of flowcharts that are important to understand to be able to write a good pseudocode. I also learned about flowcharts, how they are structured, and what the shapes used to create them represent. I also learned about the history of flowcharts, their different types, and their significance.