INTRODUCTION TO PROBLEM SOLVING METHODS AND ALGORITHM DEVELOPMENT AND IT APPLICATION IN POPULATION STUDIES

BY

ADEYEMI BLESSING FUOYE/PHD/2025/2026/0369

ASSIGNMENT SUBMITTED TO THE DEPARMENT OF DEMOGRAPHY AND SOCIAL STATISTICS, FEDERAL UNIVERSITY OF OYE-EKITI.

JUNE, 2026

INTRODUCTION

Problem solving is one of the most important skills in Computer Science. Every computer program is created to solve a particular problem. Before writing a program, the programmer must understand the problem, design an algorithm, and then implement it using a programming language.

Population studies involve collecting, analyzing, and interpreting information about human populations. Computer science provides tools and algorithms that make this process faster, more accurate, and more efficient.

WHAT IS PROBLEM SOLVING?

Problem solving is the systematic process of identifying a problem, analyzing it, and finding the most suitable solution.

It's also a process of identifying a problem, developing an algorithm for the identified problem and finally implementing the algorithm to develop a computer program.

In Computer Science, problem solving means using logical steps to create a solution that a computer can execute.

Example 1

Problem:

Governor Biodun Abayomi Oyebanji wants to know the total population of six local governments in Ekiti State.

Solution:

Collect the population of each local government and calculate the total.

Ado Ekiti	2,500
Efon	3,000
Emure	4,500
Ikere	1,800
Oye	2,200
Ikole	3,200

Total Population

= 2500 + 3000 + 4500 + 1800 + 2200+3200

= **17,000 people**

Example 2

Problem

Suppose while driving, a vehicle starts making a strange noise. We might not know how to solve the problem right away.

Solution

First, we need to identify from where the noise is coming? In case the problem cannot be solved by us, then we need to take the vehicle to a mechanic. The mechanic will analyze the problem to identify the source of the noise, make a plan about the work to be done and finally repair the

vehicle in order to remove the noise. From the above example, it is explicit that, finding the solution to a problem might consist of multiple steps.

When problems are straightforward and easy, we can easily find the solution. But a complex problem requires a methodical approach to find the right solution. In other words, we have to apply problem solving techniques. Problem solving begins with the precise identification of the problem and ends with a complete working solution in terms of a program or software.

KEY STEPS REQUIRED FOR SOLVING A PROBLEM USING A COMPUTER

i. **Problem Definition: -** Understand what needs to be solved or understand exactly what is required, what's needed and the information provided.
   Example
   Find the local government with the highest population in Ekiti State.

ii. **Analyze the Problem/Problem Analysis**: - Breaking big problems into smaller parts and identify the required inputs, the processing steps, and the expected outputs.
   Example
   Input
   Population data
   Process
   Compare all values
   Output
   Local Government with the highest population in Ekiti State.

iii. **Define Solution:-** Develop an algorithm or draw a flowchart that outlines the step-by-step solution to the problem in plain English. No programming code is required at this stage.

iv. **Code Implementation:** - Translate the algorithm into a programming language that the computer can understand by writing a computer program.

v. **Testing and Debugging:** - Run the program and to Check whether the program or output works correctly. Identify and correct any errors (bugs) found during testing.

vi. **Maintenance: -** Correct errors and improve the system.

PROBLEM SOLVING METHODS

Problems themselves can be classified into two different categories known as ill-defined and Well-defined problems (Schacter, 2009). Ill-defined problems represent issues that do not have clear goals, solution paths, or expected solutions whereas well-defined problems have specific goals, clearly defined solutions, and clear expected solutions. Problem solving often incorporates logical reasoning and interpretation of meanings behind the problem, and also in many cases require abstract thinking and creativity in order to find novel solutions. Various methods of studying problem solving exist including introspection, simulation, computer modeling, and experimentation. Understanding how a variety of problem-solving strategies work is important because different problems typically require you to approach them in different ways to find the

best solution. By mastering several problem-solving strategies, you can more effectively select the right plan of action when faced with challenges in the future. This can help you solve problems faster and develop stronger critical thinking skills.

i. **Trial and Error Method**

A trial-and-error approach to problem-solving involves trying a number of different solutions and ruling out those that do not work. This approach can be a good option if you have a very limited number of options available.

Example 1:- In terms of a broken printer for example, one could try checking the ink levels, and if that doesn't work, you could check to make sure the paper tray isn't jammed. Or maybe the printer isn't actually connected to a laptop. When using trial and error, one would continue to try different solutions until the problem is solved. Although trial and error is not typically one of the most time-efficient strategies, it is a commonly used one.

Example

Trying different passwords until the correct one works.

Summary Different solutions are tested until the correct one is found.

ii. **Algorithm Method**

A common type of strategy is an algorithm. An algorithm is a problem-solving formula that provides you with step-by-step instructions used to achieve a desired outcome (Kahneman, 2011). You can think of an algorithm as a recipe with highly detailed instructions that produce the same result every time they are performed. Algorithms are used frequently in our everyday lives, especially in computer science.

Example

When you run a search on the Internet, search engines like Google use algorithms to decide which entries will appear first in your list of results. Facebook also uses algorithms to decide which posts to display on your newsfeed.

iii. **Divide And Conquer**

A large problem is divided into smaller problems.

Example

Instead of counting Nigeria's population at once, count each state separately before adding the totals.

iv. A Heuristic Method

A heuristic Method another type of problem solving strategy. While an algorithm must be followed exactly to produce a correct result, a heuristic is a general problem-solving framework (Tversky etal., 2022). You can think of these as mental shortcuts that are used to solve problems. A "rule of thumb" is an example of a heuristic. Such a rule saves the person time and energy when making a decision, but despite its time-saving characteristics, it is not always the best method for making a rational decision. Different types of heuristics are used in different types of situations, but the impulse to use a heuristic occurs when one of five conditions is met (Pratkanis, 1989):

* When one is faced with too much information

* When the time to make a decision is limited

* When the decision to be made is unimportant

* When there is access to very little information to use in making the decision

* When an appropriate heuristic happens to come to mind in the same moment

Working backwards is a useful heuristic in which you begin solving the problem by focusing on the end result. It is common to use the working backwards heuristic to plan the events of your day on a regular basis, probably without even thinking about it. Another useful heuristic is the practice of accomplishing a large goal or task by breaking it into a series of smaller steps. Students often use this common method to complete a large research project or long essay for school.

Example 1, students typically brainstorm, develop a thesis or main topic, research the chosen topic, organize their information into an outline, write a rough draft, revise and edit the rough draft, develop a final draft, organize the references list, and proofread their work before turning in the project. The large task becomes less overwhelming when it is broken down into a series of small steps.

Example 2

Estimating next year's population using previous census data.

Summary:- Uses experience or educated guesses.

ALGORITHM DEVELOPMENT

Having understood the problem, it is time to come up with precise plan of what the computer is expected to do

Definition: Algorithm is a precise sequence of instructions for solving a problem. Some of the more complex algorithms may be considered randomized algorithms or non-deterministic algorithms where the instructions are not necessarily in sequence and may not even have a finite number of instructions.

Algorithm development is the process of designing a clear, logical, and step-by-step solution to a problem before writing a computer program. It helps programmers think through the solution carefully, making it easier to write accurate and efficient code.

Example

Planning a journey before you start driving. You first decide where you are going, the best route to take, and what you need before you begin the trip. Similarly, in programming, you first develop an algorithm before writing the code.

THE NOTION OF ALGORITHM

By definition, an algorithm is an effective step-by-step procedure for solving a problem in a finite number of steps. In other words, it is a finite set of well-defined instructions or step-by-step description of the procedure written in human readable language for solving a given problem. An algorithm itself is division of a problem into small steps which are ordered in sequence and easily understandable. Algorithms are very important to the way computers process information, because a computer program is basically an algorithm that tells computer what specific tasks to perform in what specific order to accomplish a specific task. The same problem can be solved with different methods. So, for solving the same problem, different algorithms can be designed. In these algorithms, number of steps, time and efforts may vary more or less.

REASONS FOR ALGORITHM

A programmer writes a program to instruct the computer to do certain tasks as desired. The computer then follows the steps written in the program code. Therefore, the programmer first prepares a roadmap of the program to be written, before actually writing the code. Without a roadmap, the programmer may not be able to clearly visualise the instructions to be written and may end up developing a program which may not work as expected. Such a roadmap is nothing but the algorithm which is the building block of a computer program. For example, searching using a search engine, sending a message, finding a word in a document, booking a taxi through an app, performing online banking, playing computer games, all are based on algorithms. Writing an algorithm is mostly considered as a first step to programming. Once we have an algorithm to solve a problem, we can write the computer program for giving instructions to the

computer in high level language. If the algorithm is correct, computer will run the program correctly, every time. So, the purpose of using an algorithm is to increase the reliability, accuracy and efficiency of obtaining solutions.

STEPS INVOLVED IN ALGORITHM DEVELOPMENT

An algorithm can be defined as "a complete, unambiguous, finite number of logical steps for solving a specific problem "

Step 1. Identification of input: For an algorithm, there are quantities to be supplied called input and these are fed externally. The input is to be identified first for any specified problem.

Step 2: Identification of output: From an algorithm, at least one quantity is produced, called for any specified problem.

Step 3: Identify the processing operations: All the calculations to be performed in order to lead to output from the input are to be identified in an orderly manner. 45

Step4: Processing Definiteness: The instructions composing the algorithm must be clear and there should not be any ambiguity in them.

Step 5: Processing Finiteness: If we go through the algorithm, then for all cases, the algorithm should terminate after a finite number of steps.

Step 6: Possessing Effectiveness: The instructions in the algorithm must be sufficiently basic and in practice they can be carries out easily.

STAGES OF ALGORITHM DEVELOPMENT

1. Understand the Problem

The first stage is to clearly understand what problem needs to be solved.

At this stage, ask questions such as:

- What is the problem?
- What is expected?
- What information is available?
- What result should be produced?

Example

Problem:
The department of demography and social statistics in Federal University Oye Ekiti wants to calculate the average score of 50 students.

Before writing the program, you must understand that the program should:

- Accept the scores of 50 students.
- Add all the scores together.
- Divide the total by 50.
- Display the average score.

2. Identify the Inputs

Inputs are the data or information that the algorithm requires to solve the problem.

Example

Problem: Calculate the Body Mass Index (BMI) of Mrs. Adeyemi Blessing**,** a student of the Department of Demography and Social Statistics**.**

Inputs:
- Weight (kg)
- Height (m)

3. Identify the Outputs

The **output** is the final result produced by the algorithm after processing the input data.

Problem: Determine which of the three departments has the largest population.

Inputs:

- Population of Department 1
- Population of Department 2
- Population of Department 3

Output:

- The department with the largest population

4. Design the Logical Steps

This stage involves writing the sequence of steps that will solve the problem. These steps can be written as pseudocode or drawn as a flowchart**.**

Example

Problem: Calculate the average of three numbers.

Algorithm

1. Start.
2. Input the first number.
3. Input the second number.
4. Input the third number.
5. Add the three numbers.
6. Divide the total by 3.
7. Display the average.
8. Stop.

5. Test the Algorithm

After developing the algorithm, it should be tested using sample data to verify that it produces the correct results.

Example

Suppose the algorithm calculates the average of fours numbers.

Input:

- 40
- 20
- 40

Expected Result:

Average = (40 + 20 + 40) ÷ 4

Average = 25

If the algorithm gives 25, it is working correctly.

Testing helps identify mistakes before the program is written.

CHARACTERISTICS OF ALGORITHM

An algorithm must possess following characteristics:

1. Precision — the steps are precisely stated or defined.

2. Uniqueness — results of each step are uniquely defined and only depend on the input

and the result of the preceding steps.

3. Finiteness — the algorithm always stops after a finite number of steps.

4. Input — the algorithm receives some input.

5. Output — the algorithm produces some output.

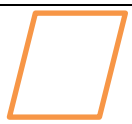
METHOD OF REPRESENTING ALGORITHMN

To develop an algorithm, the instructions must be represented in a way that is understandable to a person who is trying to figure out the steps involved. Two commonly used representations for an algorithm is by using (1) flowcharts or (2) pseudo code.

FLOW CHART

A flowchart is a graphical representation of an algorithm using standard symbols connected by arrows to show the sequence of operations.

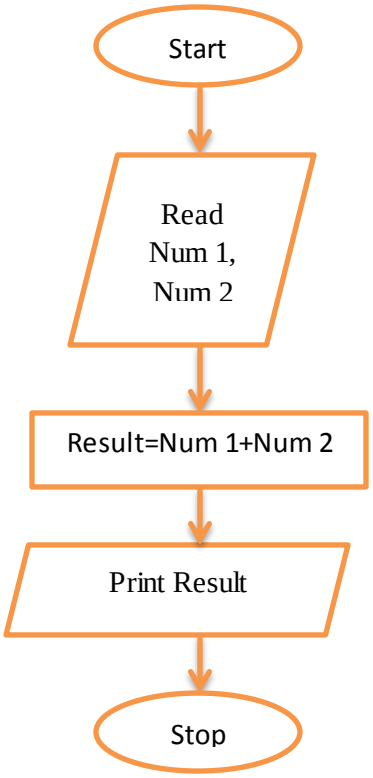
Common Flowchart Symbols

Symbol	Name	Purpose
Oval 	Start/Stop	Indicates the beginning or end of the algorithm.
Rectangle 	Process	Represents a calculation or instruction.
Parallelogram 	Input/output	Represents data input or output.

Diamond 	Decision	Represents a condition or comparison.
Arrow 	Flow Line	Shows the direction of execution.

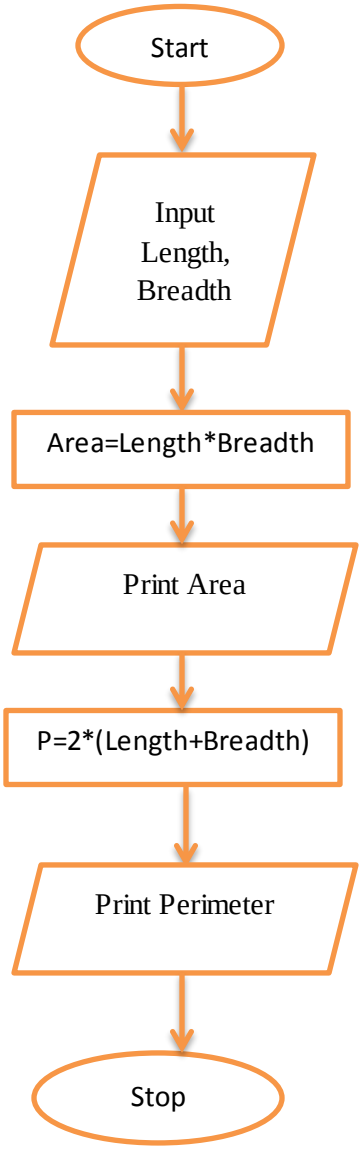
Example 1

Write an algorithm to display the sum of two numbers entered by users using flowcharts



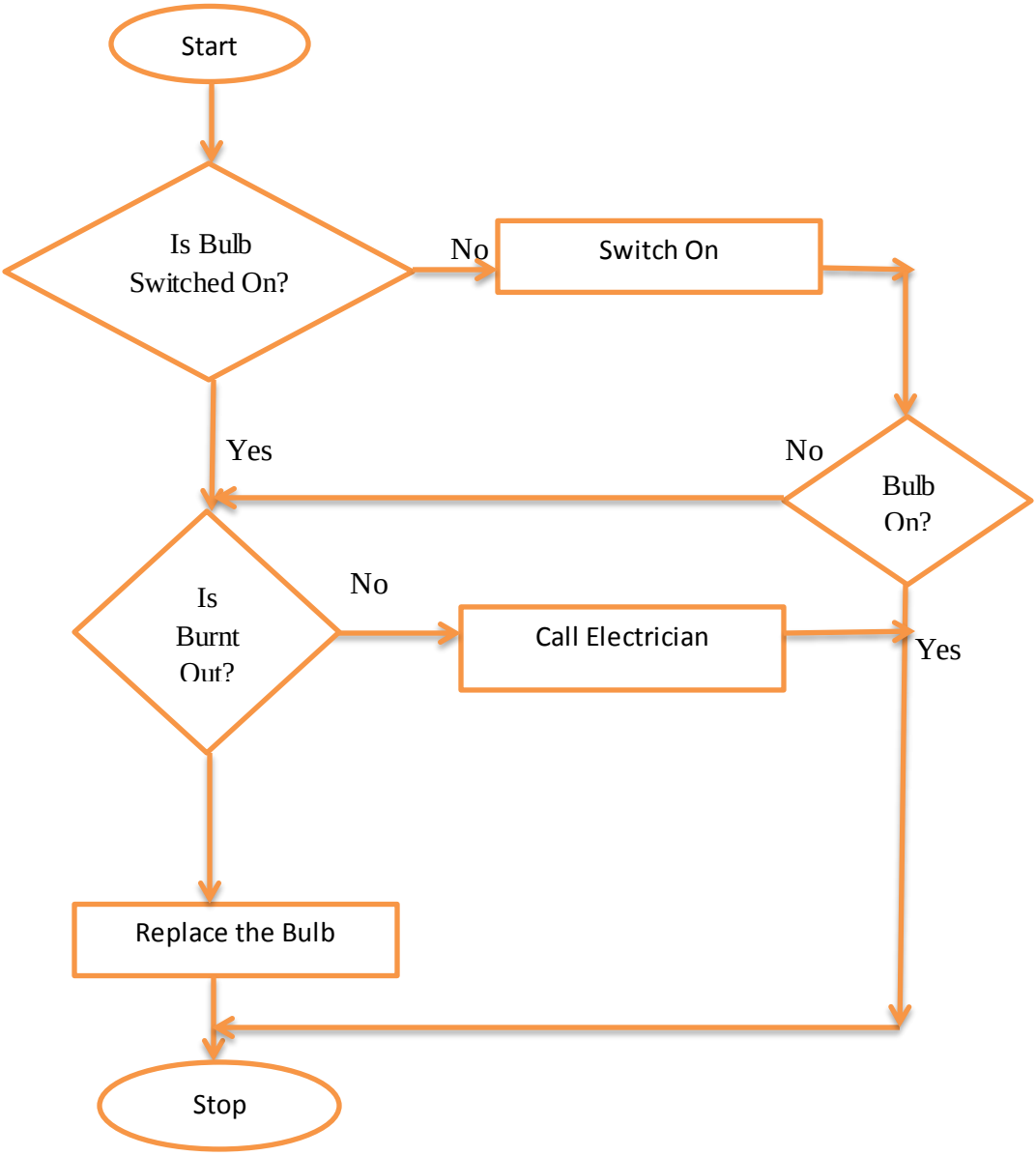
Example 2

Write an algorithm to calculate area and perimeter of rectangle using flowcharts



Example 3

Draw a flow chart to solve the problem of a non-functional light bulb



2. Pseudocode

Pseudocode is an informal way of writing an algorithm using programming-like statements without following the syntax of a specific programming language.

A pseudocode (pronounced Soo-doh-kohd) is another way of representing an algorithm. It is considered as a non-formal language that helps programmers to write algorithm. It is a detailed description of instructions that a computer must follow in a particular order. It is intended for human reading and cannot be executed directly by the computer. No specific standard for writing a pseudocode exists. The word "pseudo" means "not real," so "pseudocode" means "not real code".

Following are some of the frequently used keywords while writing pseudocode:

* INPUT

* COMPUTE

* PRINT

* INCREMENT

* DECREMENT

* IF/ELSE

* WHILE

* TRUE/FALSE

Advantages

- Easy to convert into a computer program.
- Independent of any programming language.
- Easy to understand and modify.

Disadvantages

- No universal standard format.
- Cannot be executed directly by a computer.

Example 1

Write an algorithm to calculate area and perimeter of a rectangle, using pseudocode.

Pseudocode for calculating area and perimeter of a rectangle.

```
INPUT length
INPUT breadth
COMPUTE Area = length * breadth
PRINT Area
COMPUTE Perim = 2 * (length + breadth)
PRINT Perim
```

Example 2

Write an algorithm to display the sum of two numbers entered by user using pseudocode.

Pseudocode for the sum of two numbers will be:

```
INPUT num1
INPUT num2
COMPUTE Result = num1 + num2
PRINT Result
```

Example 3

Write pseudocode that tells a user that the number they entered is not a 5 or a 6.

```
READ is five
IF (is five = 5)
WRITE "your number is 5"
ELSE IF (is five = 6)
WRITE "your number is 6"
ELSE
```

WRITE "your number is not 5 or 6"

END IF

APPLICATION OF ALGORITHMS IN POPULATION STUDIES

Population Studies, also known as Demography, i**s the scientific study of human** populations. It involves collecting, analyzing, interpreting, and presenting data about population size, structure, growth, distribution, birth rates, death rates, migration patterns, and other demographic characteristics. As populations continue to increase around the world, especially in developing countries such as Nigeria, huge amounts of demographic data are generated through population censuses, household surveys, hospital records, and national registration systems.

Processing such enormous datasets manually is time-consuming, expensive, and prone to human error. Computer Science provides powerful tools, algorithms, and software applications that enable researchers, governments, and organizations to process population data quickly and accurately. Algorithms make it possible to predict future population growth, estimate migration patterns, calculate population density, monitor disease outbreaks, and support effective planning for schools, hospitals, roads, housing, employment, and other social services.

Therefore, the integration of Computer Science and Population Studies has greatly improved demographic research by making data collection, analysis, forecasting, and decision-making more efficient, reliable, and cost-effective.

Algorithms play an essential role in modern demographic analysis. They help researchers and government's process large datasets accurately and efficiently.

a. Population Census

Algorithms are used to organize, validate, and summarize census data collected from households.

b. Population Growth Prediction

Using mathematical models and historical census records, algorithms estimate future population sizes.

c. Birth Rate Analysis

Birth Rate = (Number of Live Births ÷ Total Population) × 1,000

It helps governments monitor fertility trends and plan maternal and child healthcare services.

d. Death Rate Analysis

Algorithms calculate mortality rates to evaluate public health conditions and identify areas requiring healthcare interventions.

e. Migration Analysis

Algorithms analyze the movement of people into and out of regions, helping governments understand urbanization, labor migration, and population redistribution.

f. Population Density Calculation

Population Density = Population ÷ Land Area

Example:

Population = 80,000

Land Area = 20 km^2

Density = 80,000 ÷ 20 = **4,000 people per km^2**

Population density information assists in urban planning, housing development, and infrastructure provision.

g. Resource Allocation

Governments use algorithms to determine where to build schools, hospitals, roads, and water supply systems based on population size and distribution.

h. Disease Surveillance and Public Health

Algorithms analyze demographic and health data to detect disease outbreaks, identify high-risk populations, and support vaccination campaigns.

i. Policy Formulation and Decision-Making

Demographic algorithms provide evidence-based information that helps policymakers design effective social and economic development strategies.

j. Poverty and Employment Analysis

Algorithms help analyze unemployment rates, income distribution, and poverty levels, enabling governments to implement targeted intervention programs.

Examples of Population Problems Solved Using Algorithms

Algorithms are used to solve many real-world problems in population studies. Some examples include:

Example 1: Calculating Population Growth

A demographer wants to estimate the population of Ado Ekiti, Ekiti State after one year.

Given:

- Current population in Ado Ekiti = 50,000
- Annual growth rate in Ado Ekiti = 3%

Formula:

Future Population = Current Population × (1 + Growth Rate)

Future Population = 50,000 × (1 + 0.03)

Future Population = 51,500

Using this algorithm, government agencies can estimate future population sizes for planning purposes.

Pseudocode

BEGIN

INPUT CurrentPopulation

INPUT GrowthRate

FuturePopulation = CurrentPopulation × (1 + GrowthRate)

OUTPUT FuturePopulation

END

Example 2: Identifying the Most Populated State

Suppose the population of four states is as follows:

- Osun = 4,500,000
- Ekiti = 6,700,000
- Oyo = 5,200,000
- Lagos = 8,100,000

Algorithm:

1. Input the population of all states.
2. Compare the values.
3. Select the highest value.
4. Display the corresponding state.

Output:

Lagos State has the highest population.

```
BEGIN
 INPUT OsunPopulation
  INPUT EkitiPopulation
   INPUT OyoPopulation
   INPUT LagosPopulation
 LargestPopulation ← OsunPopulation
    LargestState ← "Osun"
 IF EkitiPopulation > LargestPopulation THEN
 LargestPopulation ← EkitiPopulation
LargestState ← "Ekiti"
    ENDIF
```

```
    IF OyoPopulation > LargestPopulation THEN
        LargestPopulation ← OyoPopulation
        LargestState ← "Oyo"
    ENDIF

    IF LagosPopulation > LargestPopulation THEN
        LargestPopulation ← LagosPopulation
        LargestState ← "Lagos"
    ENDIF
OUTPUT LargestState, "State has the highest population."
END
```

Example 3: Calculating Population Density of Oye Local Government

Population Density = Population ÷ Land Area

Given:

- Population = 206,000
- Land Area = 50 km²

Density = 250,000 ÷ 50

Density = **4,120 people per km²**

This information helps governments identify overcrowded areas.

Pseudocode

```
BEGIN
  INPUT Population
    INPUT LandArea
  PopulationDensity ← Population / LandArea
OUTPUT "Population Density = ", PopulationDensity, " people per km²"
END
```

Example 4: Calculate Birth Rate In Ikere Local government

Birth Rate = (Number of Live Births ÷ Total Population) × 1,000

Suppose:

- Live Births = 2,400
- Population = 80,000

Birth Rate = (2,400 ÷ 80,000) × 1,000

Birth Rate = **30 births per 1,000 population**

Pseudocode

```
BEGIN
  INPUT LiveBirths
    INPUT TotalPopulation
 BirthRate ← (LiveBirths / TotalPopulation) × 1000
OUTPUT "Birth Rate = ", BirthRate, " births per 1,000 population"
END
```

Example 5: Calculate Death Rate in Ikole Local Government

Death Rate = (Number of Deaths ÷ Total Population) × 1,000

Suppose:

- Deaths = 600
- Population = 120,000

Death Rate = (600 ÷ 120,000) × 1,000

Death Rate = **5 deaths per 1,000 population**.

Pseusocode

BEGIN

INPUT NumberOfDeaths

INPUT TotalPopulation

DeathRate ← (NumberOfDeaths / TotalPopulation) × 1000

OUTPUT "Death Rate = ", DeathRate, " deaths per 1,000 population"

END

Conclusion

Problem solving and algorithm development are fundamental concepts in Computer Science that provide systematic approaches to solving complex real-world problems. Algorithms enable computers to process information efficiently, accurately, and consistently, making them indispensable in software development and data analysis.

In Population Studies, algorithms have become essential tools for processing census data, estimating population growth, calculating demographic indicators such as birth and death rates, monitoring migration, determining population density, and supporting evidence-based planning. They help governments and researchers make informed decisions regarding education, healthcare, housing, transportation, employment, and national development.

Although challenges such as poor data quality, inadequate infrastructure, and privacy concerns exist, the benefits of algorithmic approaches far outweigh these limitations. As technology

continues to advance, the application of algorithms, artificial intelligence, and big data analytics in population studies will become increasingly important for sustainable development and effective governance.

References

Aho, A. V., Lam, M. S., Sethi, R., & Ullman, J. D. (2007). *Compilers: Principles, Techniques, and Tools* (2nd ed.). Pearson.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2022). *Introduction to Algorithms* (4th ed.). MIT Press.

Knuth, D. E. (1997). *The Art of Computer Programming* (Vol. 1). Addison-Wesley.

Weiss, M. A. (2014). *Data Structures and Algorithm Analysis in C++* (4th ed.). Pearson.

Shryock, H. S., Siegel, J. S., & Associates. (1976). *The Methods and Materials of Demography*. Academic Press.

United Nations. (2022). *World Population Prospects 2022*. United Nations Department of Economic and Social Affairs.

National Population Commission (Nigeria). (2006). *Population and Housing Census of the Federal Republic of Nigeria*. Abuja: NPC.

Preston, S. H., Heuveline, P., & Guillot, M. (2001). *Demography: Measuring and Modeling Population Processes*. Blackwell Publishing.