Republic of the Philippines
Department of Education
REGION III
SCHOOLS DIVISION OFFICE OF NUEVA ECIJA

LEARNING ACTIVITY SHEET
SPECIAL PROGRAM IN ICT 9
BASIC PROGRAMMING 9
First Quarter, Week 4

Name of Learner: _____

Grade Level & Section: _____ Date: _____

Algorithm

BACKGROUND INFORMATION FOR LEARNERS

Would you be able to think about a day in your life which goes without problem solving?

In our everyday activity, for example, buying something from an overall store and making instalments, keeping expense in school, or pulling back cash from financial balance. Every one of these exercises includes a problem solving. It tends to be said that whatever activity a person do to acquire specific objective or goal comes under problem solving.

In our day to day activity such as purchasing something from a general store and making payments, depositing fee in school, or withdrawing money from bank account. All these activities involve some kind of problem solving. It can be said that whatever activity a human being or machine do for achieving a specified objective comes under problem solving. To make it clearer, let us see some other examples.

Example 1

If you are watching a news channel on your TV and you want to change it to a sports channel, you need to do something i.e. move to that channel by pressing that channel number on your remote. This is a kind of problem solving.

Example 2

One Monday morning, a student is ready to go to school but yet he/she has not picked up those books and copies which are required as per timetable. So here picking up books and copies as per timetable is a kind of problem solving.

Example 3

If someone asks to you, what is time now? So seeing time in your watch and telling him is also a kind of problem solving.

If you can solve a given problem then you can also write an algorithm for it. In next section we will learn what an algorithm is.

Programming is a process of problem solving. Analyzing the problem, outline the problem requirements and designing steps (algorithm) are some common techniques in problem solving.

Algorithm is the idea behind the computer program. It stays the same independent of which kind of hardware it is running on and which programming language it is written in. It solves a well-specified problem in a general way by describing the set of instances (input) it must work on and describing the desired properties of the output. Before a computer can perform a task, it must have an algorithm that tells it what to do. Informally: "An algorithm is a set of steps that define how a task is performed". Formally: "An algorithm is an ordered set of unambiguous executable steps, defining a terminating process".

- Ordered set of steps: structure!
- Executable steps: doable!
- Unambiguous steps: follow the directions!
- Terminating: must have an end!

IMPORTANT PROPERTIES OF ALGORITHMS

- Correct. Always returns the desired output for all legal instances of the problem.
- Unambiguous
- Precise
- Efficient. Can be measured in terms of time and space. Time tends to be more important

While writing algorithms we will use following symbol for different operations:

'+' for Addition

'-' for Subtraction

'*' for Multiplication

'/' for Division and

' ' for assignment. For example A X*3 means A will have a value of X*3.

REPRESENTATION OF ALGORITHMS

A single algorithm can be represented in many ways:

- Formulas: F = (9/5)C + 32
- Words: Multiply the Celsius by 9/5 and add 32.
- Flow Charts.
- Pseudo-code. Pseudocode is like a programming language but its rules are less strict.

In each case, the algorithm stays the same; the implementation differs.

A program is a representation of an algorithm designed for computer applications. The process is the activity of executing a program, or executes the algorithm represented by the program.

STEPS IN WRITING AN ALGORITHM IN PROGRAMMING

1. Determine the outcome of your code. What is the specific problem you want to solve or the task you want it to accomplish?
2. Decide on a starting point. Finding your starting and ending point are crucial to listing the steps of the process. To determine a starting point, determine the answers to these questions:
   - What data/inputs are available?
   - Where is that data located?
   - What formulas are applicable to the issue at hand?
   - What are the rules to working with the available data?
   - How do the data values relate to each other?
3. Find the ending point of the algorithm. As with the starting point, you can find the end point of your algorithm by focusing on these questions:
   - What facts will we learn from the process?
   - What changes from the start to the end?
   - What will be added or no longer exist?
4. List the steps from start to finish. Start with broad steps. To use a real-world example, let's say your goal is to have lasagna for dinner. You've determined that the starting point is to find a recipe, and that the end result is that you'll have lasagna fully cooked and ready to eat by 7 PM.

5. Determine how you will accomplish each step. Now that you have a step by-step outline, it's time to think about how you might code each step.

6. Review the algorithm. Now that you've written your algorithm, it's time to evaluate the process. Your algorithm is designed to accomplish something specific, and you'll need it to start writing your program.

Here is the algorithm for going to the market to purchase a pen.

1. Get dressed to go the market.
2. Check your wallet for money.
3. If there is no money in the wallet, replenish it.
4. Go to the shop.
5. Ask for your favorite brand of pen.
6. If pen is not available, go to step 7 else go to step 10.
7. Give money to the shopkeeper
8. Put the purchased pen on your bag.
9. Go back home
10. Ask for any other brand of pen.
11. Go to step 7.

EXAMPLE OF ALGORITHM

Problem 1: Find the area of a Circle of radius r.

Inputs to the algorithm: Radius r of the Circle.
Expected output: Area of the Circle
Algorithm:
Step 1: Read\input the Radius r of the Circle
Step 2: Area PI*r*r // calculation of area
Step 3: Print Area

Problem 2: Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm:
First num1.
Second num2.

Expected output:

Sum of the two numbers.

Algorithm:

Step1: Start

Step 2: Read\input the first num1.

Step 3: Read\input the second num2.

Step 4: Sum num1+num2 // calculation of sum

Step 5: Print Sum

Step 6: End

Problem 3: Write an algorithm to find the greater number between two numbers

Inputs to the algorithm:

First num1.

Second num2.

Expected output:

Print greater number

Algorithm:

Step1: Start

Step2: Read/input num1 and num2

Step3: If num1 greater than num2 then greater number= num1

Step4: if num2 greater than num1then greater number = num2

Step5: Print greater number

Step6: End

LEARNING COMPETENCY

Define and create an algorithm.

ACTIVITIES

ACTIVITY 1

Directions: Watch the video using the link below that explains what algorithm is.

https://www.youtube.com/watch?v=kM9ASKAni_s
"Computer Science Basics: Algorithms"

ACTIVITY 2

Directions: Read the sentence carefully. Write on the space provided the word “true” if the statement is correct and “false” if the statement is incorrect.

_____1. Algorithm is a step-by-step instructions used to solve problem.

_____2. An algorithm is an ordered set of unclear executable steps.

_____3. A set of directions is the algorithm we use when we're looking for the Post Office.

_____4. A set of directions is the algorithm we use when we're building a wardrobe.

_____5. Only top software programmers can write an algorithm.

_____6. A recipe is the algorithm we use when we're making a cake.

_____

_____7. When you write an algorithm the order of the instructions is very important.

_____ 8. Algorithms can be used anytime to design solutions to problems.

_____9. More than one way of solving problem should be considered when designing an algorithm.

_____10. A program is an illustration of an algorithm designed for computer applications

ACTIVITY 3

Directions: Do the following on how to create algorithm:

1.Create an algorithm to check whether the grade is passed or failed.

2.Problem3: An algorithm to find the largest number of any three numbers.

3.Create an algorithm to check whether a number is positive or negative.

REFLECTION:

What are the advantages of the algorithm and why should we use the algorithm in programming?

_____

_____

_____

_____