1. **Apply the algorithm framework for describing and solving a problem.**

The algorithm framework is a general process for describing and solving a problem. It consists of several steps that can be followed in order to identify, analyse, and solve a problem (Chuang & Wu, 2000).

Based on the ideas of Littler (2020), here is a step-by-step procedure for defining the problem and generating great ideas:

1. **Define the problem**: The first step is to define the problem you are trying to solve clearly. This involves understanding what the problem is, what the desired outcome is, and what constraints or limitations exist.
2. **Analyse the problem**: The second step is to analyse the problem in greater detail. This involves breaking the problem into its components and understanding how they are related.
3. **Develop a solution**: The third step is to develop a solution to the problem. This involves using the information gathered during the analysis phase to develop a viable solution.
4. **Implement the solution**: The fourth step is to implement the solution. This involves putting the plan into action and executing it.
5. **Test the solution**: The fifth step is to test the solution. This involves evaluating the results of the solution and determining whether it meets the desired outcome.
6. **Refine the solution**: The sixth step is to refine the solution. This involves adjusting and improving the solution based on the testing phase results.

7. **Communicate the solution**: The final step is communicating the solution. This involves sharing the solution with others who may be affected by it or who may benefit from it.

By following these steps, you can work through a problem systematically and increase your chances of finding a successful solution.

2. Demonstrate a step-by-step approach to solving a problem using pseudocode.

Writing a program in pseudocode is a language-independent programming technique that describes the function in a high-level, plain language (natural language). Pseudocode is not just for inexperienced programmers but also a fantastic problem-solving tool, especially for writing complex algorithms. It is also an excellent way to follow up on well-thought-out user scenarios and user stories. It is an excellent method for identifying ambiguous decisions and hidden side effects and defining all inputs, outputs, and interactions required to solve a problem effectively (Rogers, 2016).

According to Ubah (2021), here are some steps to solving programming problems with pseudocode:

1. Learn what the function does.
2. Check that you understand the question.
3. Dissect the problem
4. List the steps you need to take to solve the problem.
5. Convert your steps into pseudocode.
6. Create your code.

Here is an example of pseudocode written for a simple problem:

#Make a variable called sum.

#Add the first and second inputs using the addition operator.

#Assign the sum variable the result of the addition operation.

#Print out the sum variable's value.

```
DEFINE addNumbers(num1, num2)
        sum = 0
        sum = num1 + num2
        PRINT sum
ENDDEFINE
```

In this pseudocode, we define the function **addNumbers**, which accepts two arguments, **num1** and **num2.** The function then sets the variable **sum** to zero, adds **num1** and **num2** with the addition operator, and assigns the result to the variable **sum**. Finally, the function uses the **PRINT** statement to print the **sum** value.

It should be noted that this pseudocode is only an outline of the steps required to solve the problem and does not include specific syntax for any programming language.

3. Validate the output using different quality assurance matrices.

Software Quality Assurance (SQA) is a process that ensures that all software engineering processes, methods, activities, and work items are monitored and meet the defined standards (Software Testing Help, 2019).

Assume we have a programme that computes the average of a set of numbers; Galin (2004) identified the following quality assurance matrices that could be used to validate the output:

1. Functional testing involves comparing the program's functionality to the requirements. For example, we could see if the programme correctly computes the average of a list of numbers and how it handles different types of input (such as empty lists or non-numeric input).
2. Individual units or components of the programme, such as functions or methods, are tested during unit testing. For example, we could test the average function to ensure it returns the correct value for a given input.
3. Integration testing examines how various components of the programme interact. We could, for example, test whether the programme correctly integrates the function that calculates the average with other functions or modules.
4. Regression testing entails re-testing the programme after changes have been made to ensure that no new errors or bugs have been introduced. If we change the function that calculates the average, we should run regression tests to ensure that the programme still calculates the average correctly for all input types.

5. Acceptance testing entails comparing the programme to user acceptance criteria to ensure that it meets the needs and requirements of the user. We could, for example, have users test the programme to ensure that it calculates the average correctly for their specific use case.

These are only a few examples of quality assurance matrices that can be used to validate programme output. We can ensure that the programme is functional, reliable, and meets the user's needs by combining different matrices.

References:

Chuang, A.S. & Wu, F., 2000. An extensible genetic algorithm framework for problem solving in a common environment. *IEEE Transactions on power systems*, *15*(1), pp.269-275.

Littler, T. (2020). *A Framework to Solve any Problem*. [online] Medium. Available at: https://tomlittler.medium.com/a-framework-to-solve-any-problem-24692410cc37 [Accessed 27 Apr. 2023].

Rogers, K. (2016). *Using Pseudocode to Solve Complex Problems*. [online] BigThinking.io. Available at: https://bigthinking.io/using-pseudocode-to-solve-complex-problems/ [Accessed 27 Apr. 2023].

Ubah, K. (2021). *What is Pseudocode? How to Use Pseudocode to Solve Coding Problems*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/what-is-pseudocode-in-programming/.

Software Testing Help (2019). *What is Software Quality Assurance (SQA): A Guide for Beginners*. [online] Available at: https://www.softwaretestinghelp.com/software-quality-assurance/.

Galin, D., 2004. *Software quality assurance: from theory to implementation*. Pearson education.