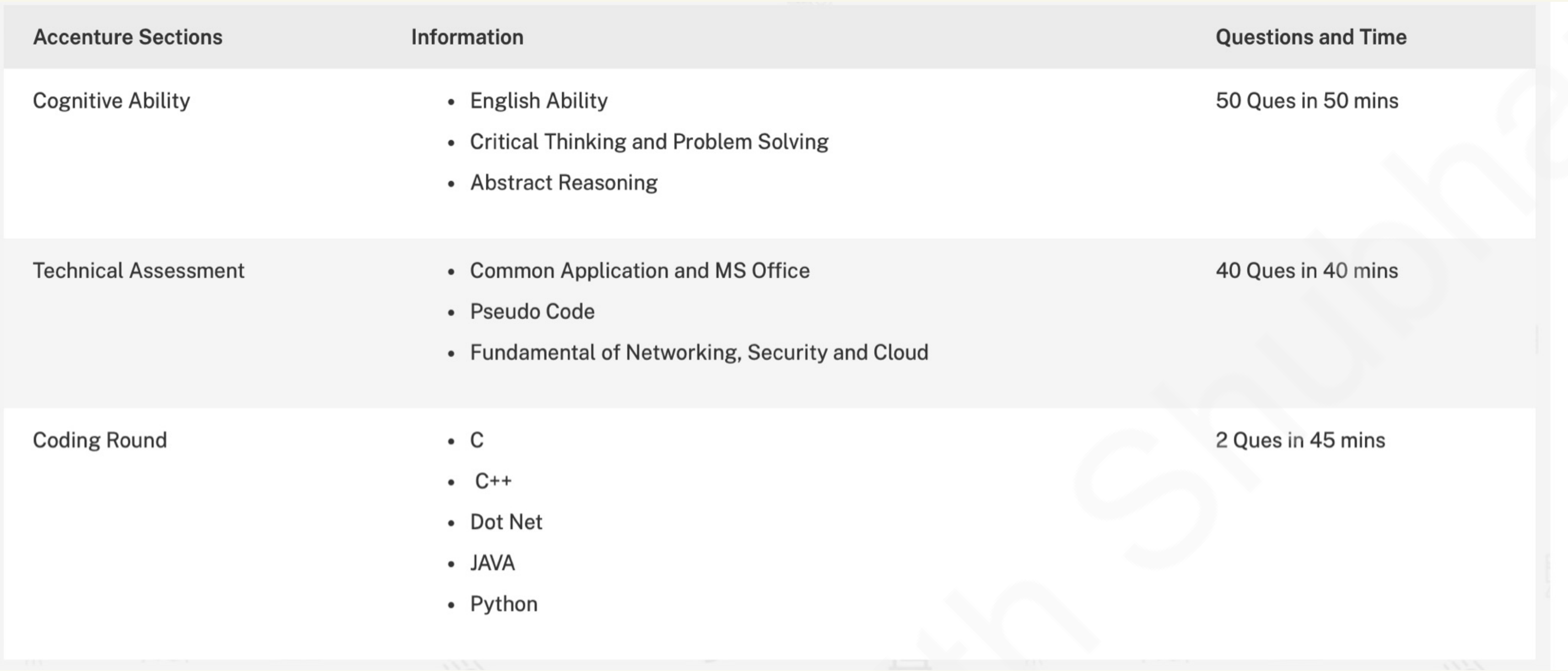


Accenture Sections	Information	Questions and Time
Cognitive Ability	• English Ability • Critical Thinking and Problem Solving • Abstract Reasoning	50 Ques in 50 mins
Technical Assessment	• Common Application and MS Office • Pseudo Code • Fundamental of Networking, Security and Cloud	40 Ques in 40 mins
Coding Round	• C • C++ • Dot Net • JAVA • Python	2 Ques in 45 mins

DEBUG WITH SHUBHAM

Accenture Technical Assessment Detailed Overview

27-sep-2024 Coding Interview Questions

https://www.youtube.com/@DebugWithShubham

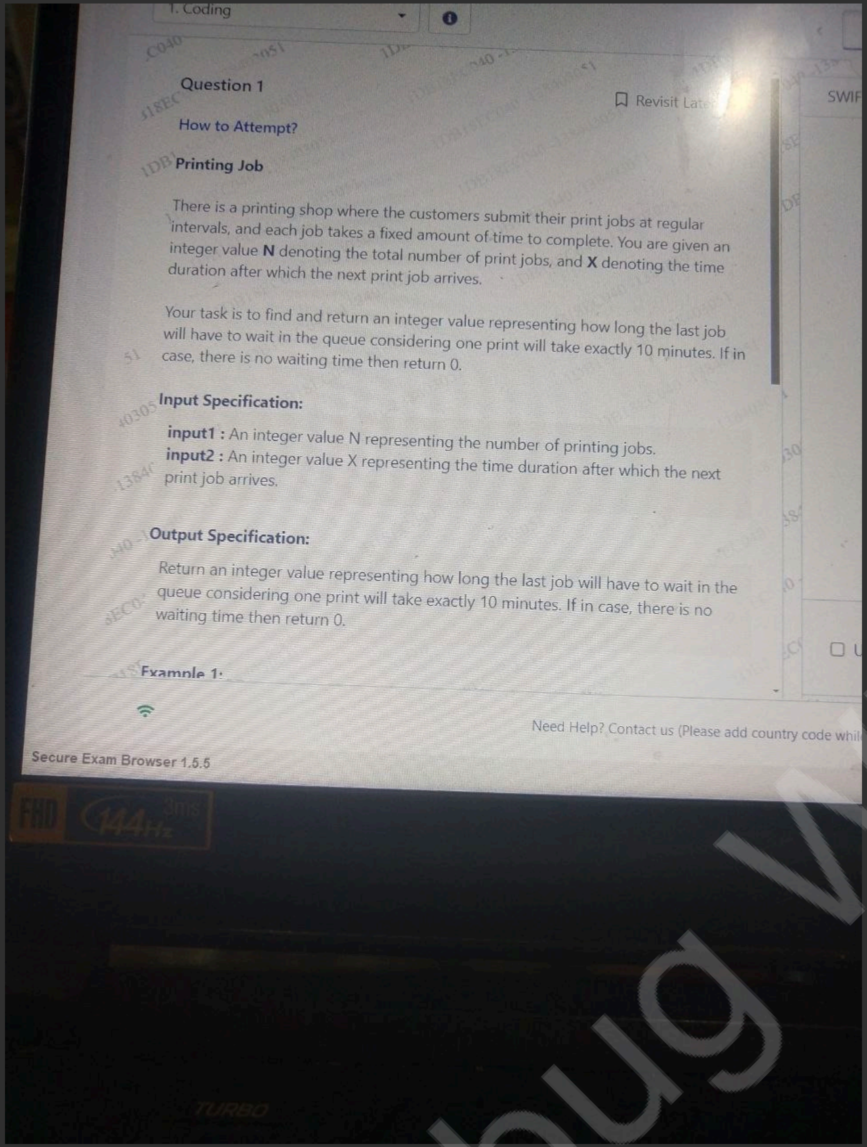
https://www.linkedin.com/in/debugwithshubham/

https://www.instagram.com/debugwithshubham/

https://topmate.io/debugwithshubham

https://t.me/debugwithshubham

Question-1



Question 1

How to Attempt?

Printing Job

There is a printing shop where the customers submit their print jobs at regular intervals, and each job takes a fixed amount of time to complete. You are given an integer value **N** denoting the total number of print jobs, and **X** denoting the time duration after which the next print job arrives.

Your task is to find and return an integer value representing how long the last job will have to wait in the queue considering one print will take exactly 10 minutes. If in case, there is no waiting time then return 0.

Input Specification:

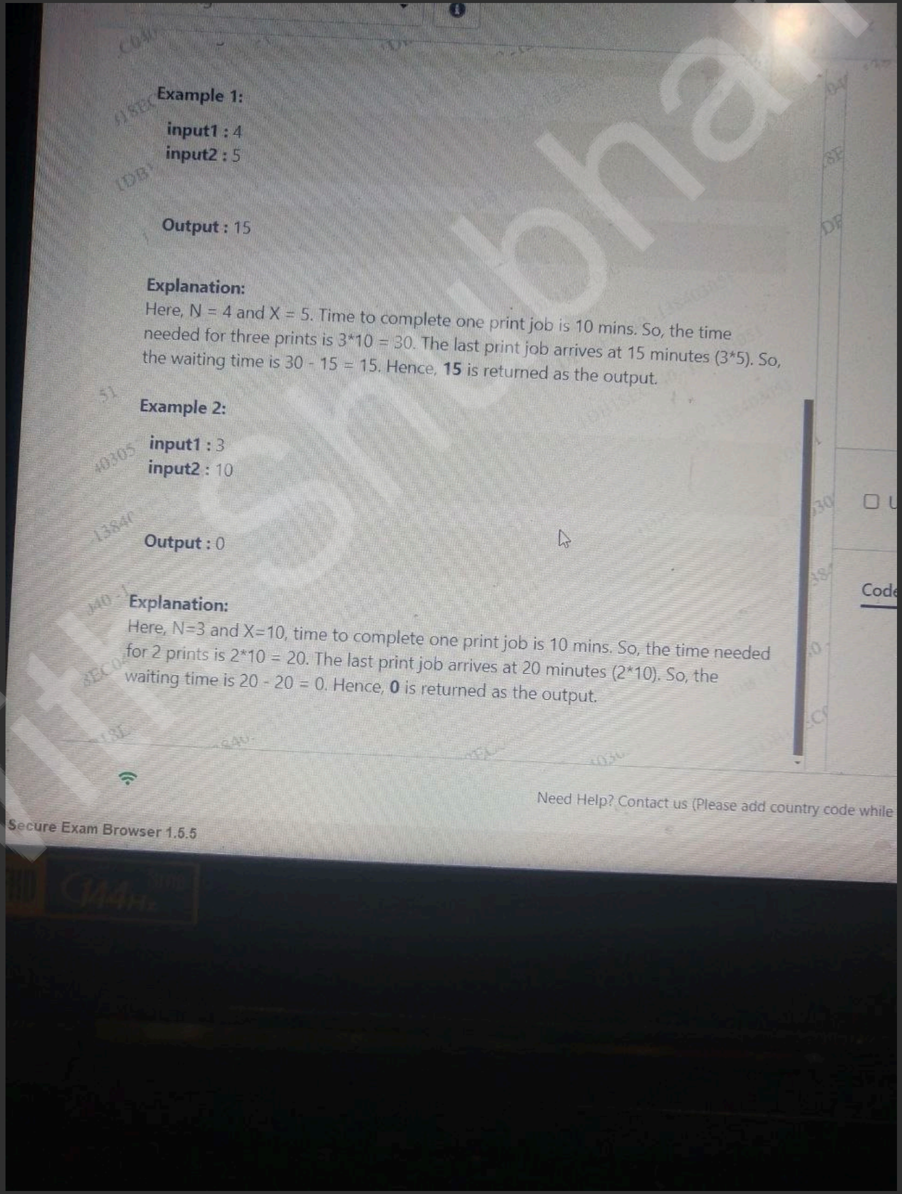
input1 : An integer value N representing the number of printing jobs.
input2 : An integer value X representing the time duration after which the next print job arrives.

Output Specification:

Return an integer value representing how long the last job will have to wait in the queue considering one print will take exactly 10 minutes. If in case, there is no waiting time then return 0.

Example 1:

Need Help? Contact us (Please add country code whil
Secure Exam Browser 1.5.5



Example 1:

input1 : 4
input2 : 5

Output : 15

Explanation:
Here, N = 4 and X = 5. Time to complete one print job is 10 mins. So, the time needed for three prints is 3*10 = 30. The last print job arrives at 15 minutes (3*5). So, the waiting time is 30 - 15 = 15. Hence, **15** is returned as the output.

Example 2:

input1 : 3
input2 : 10

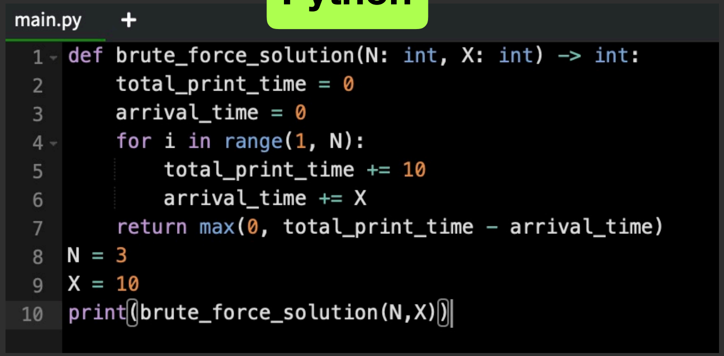
Output : 0

Explanation:
Here, N=3 and X=10, time to complete one print job is 10 mins. So, the time needed for 2 prints is 2*10 = 20. The last print job arrives at 20 minutes (2*10). So, the waiting time is 20 - 20 = 0. Hence, **0** is returned as the output.

Code
Need Help? Contact us (Please add country code while
Secure Exam Browser 1.5.5

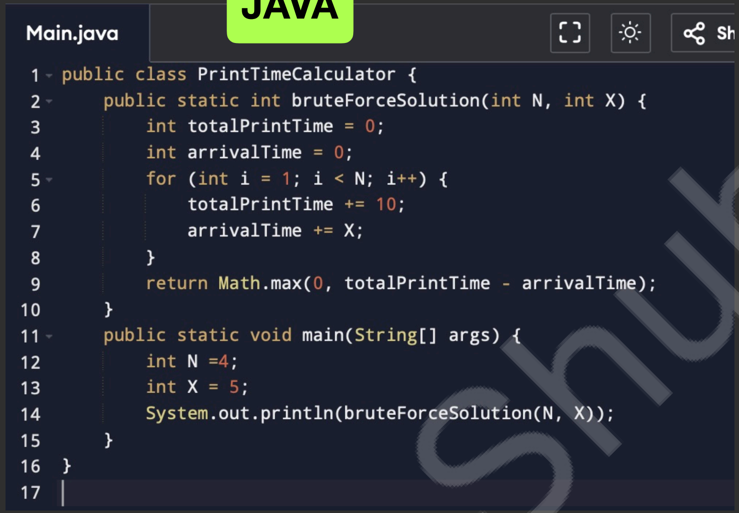
Brute Force

Python



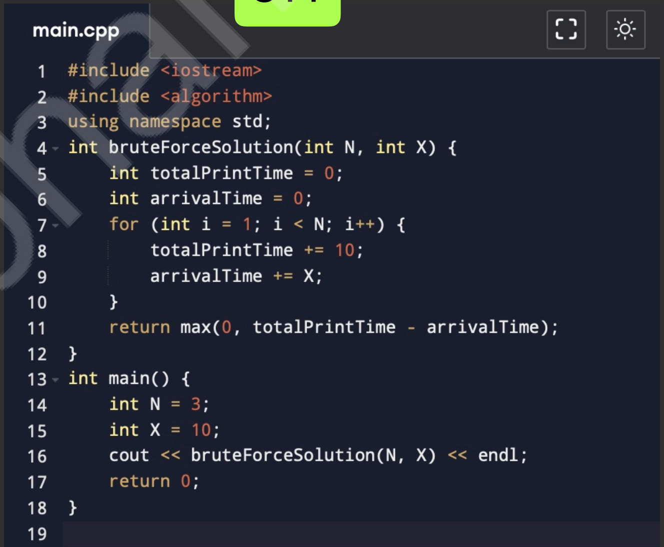
```python
def brute_force_solution(N: int, X: int) -> int:
    total_print_time = 0
    arrival_time = 0
    for i in range(1, N):
        total_print_time += 10
        arrival_time += X
    return max(0, total_print_time - arrival_time)
N = 3
X = 10
print(brute_force_solution(N,X))
```

JAVA



```java
public class PrintTimeCalculator {
    public static int bruteForceSolution(int N, int X) {
        int totalPrintTime = 0;
        int arrivalTime = 0;
        for (int i = 1; i < N; i++) {
            totalPrintTime += 10;
            arrivalTime += X;
        }
        return Math.max(0, totalPrintTime - arrivalTime);
    }
    public static void main(String[] args) {
        int N =4;
        int X = 5;
        System.out.println(bruteForceSolution(N, X));
    }
}
```

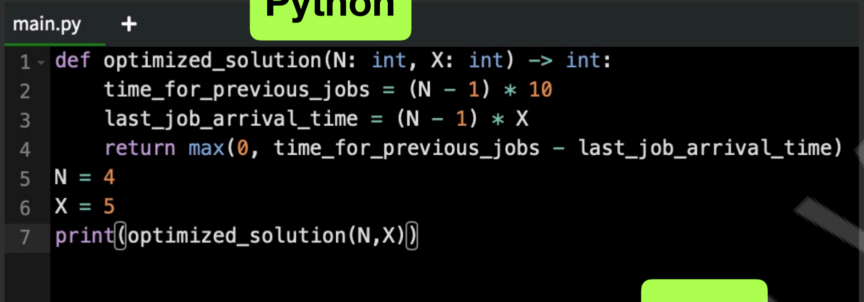
C++



```cpp
#include <iostream>
#include <algorithm>
using namespace std;
int bruteForceSolution(int N, int X) {
    int totalPrintTime = 0;
    int arrivalTime = 0;
    for (int i = 1; i < N; i++) {
        totalPrintTime += 10;
        arrivalTime += X;
    }
    return max(0, totalPrintTime - arrivalTime);
}
int main() {
    int N = 3;
    int X = 10;
    cout << bruteForceSolution(N, X) << endl;
    return 0;
}
```

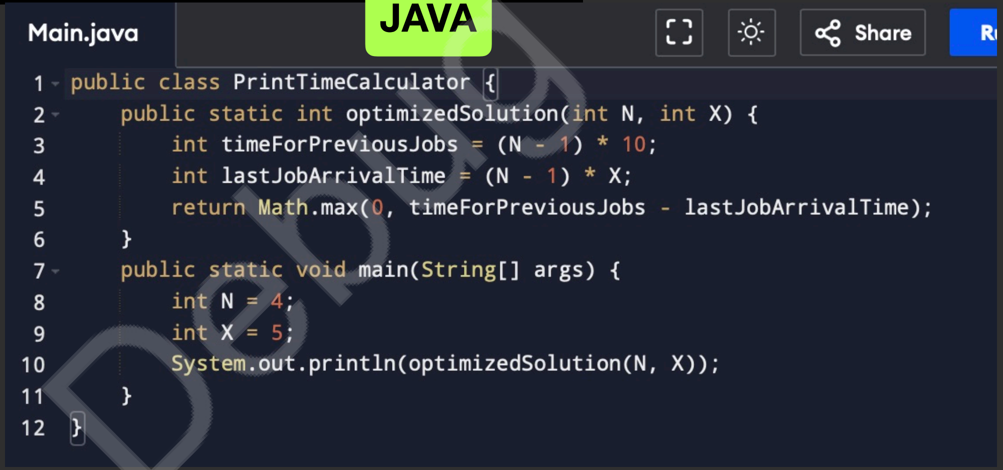
**Optimized Solution **

Python



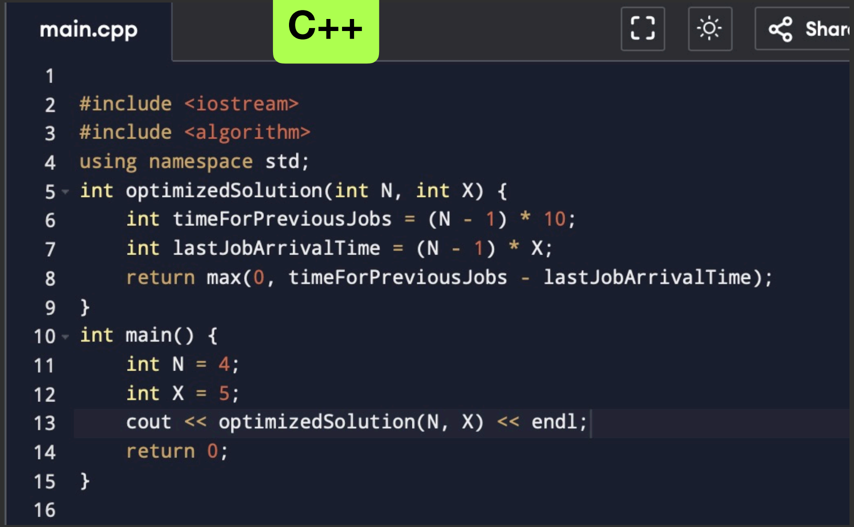
```python
def optimized_solution(N: int, X: int) -> int:
    time_for_previous_jobs = (N - 1) * 10
    last_job_arrival_time = (N - 1) * X
    return max(0, time_for_previous_jobs - last_job_arrival_time)
N = 4
X = 5
print(optimized_solution(N,X))
```

JAVA



```java
public class PrintTimeCalculator {
    public static int optimizedSolution(int N, int X) {
        int timeForPreviousJobs = (N - 1) * 10;
        int lastJobArrivalTime = (N - 1) * X;
        return Math.max(0, timeForPreviousJobs - lastJobArrivalTime);
    }
    public static void main(String[] args) {
        int N = 4;
        int X = 5;
        System.out.println(optimizedSolution(N, X));
    }
}
```

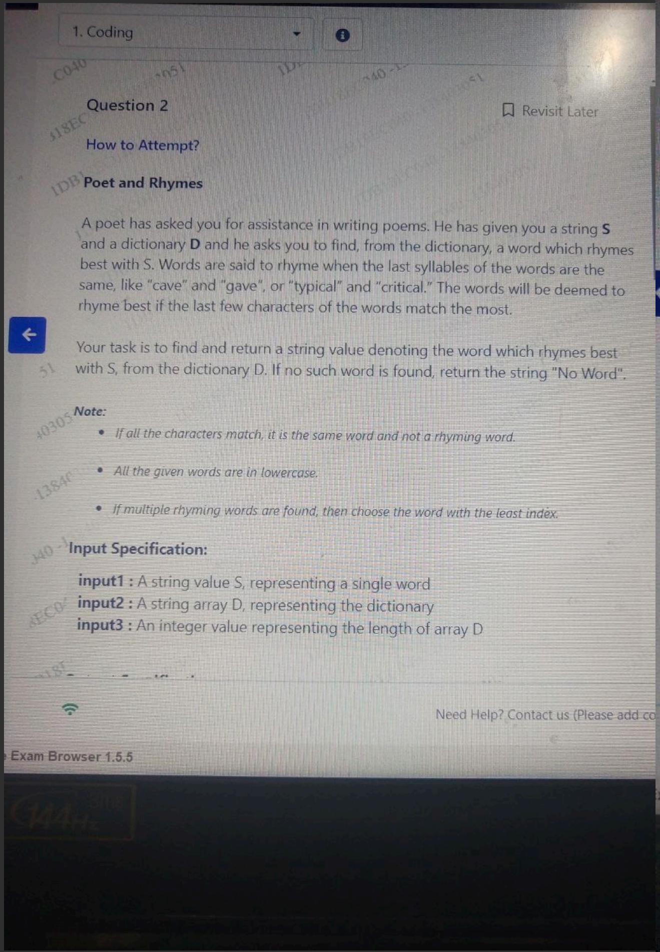
C++



```cpp

#include <iostream>
#include <algorithm>
using namespace std;
int optimizedSolution(int N, int X) {
    int timeForPreviousJobs = (N - 1) * 10;
    int lastJobArrivalTime = (N - 1) * X;
    return max(0, timeForPreviousJobs - lastJobArrivalTime);
}
int main() {
    int N = 4;
    int X = 5;
    cout << optimizedSolution(N, X) << endl;
    return 0;
}
```

Question-2



Question 2

How to Attempt?

Poet and Rhymes

A poet has asked you for assistance in writing poems. He has given you a string **S** and a dictionary **D** and he asks you to find, from the dictionary, a word which rhymes best with S. Words are said to rhyme when the last syllables of the words are the same, like "cave" and "gave", or "typical" and "critical." The words will be deemed to rhyme best if the last few characters of the words match the most.

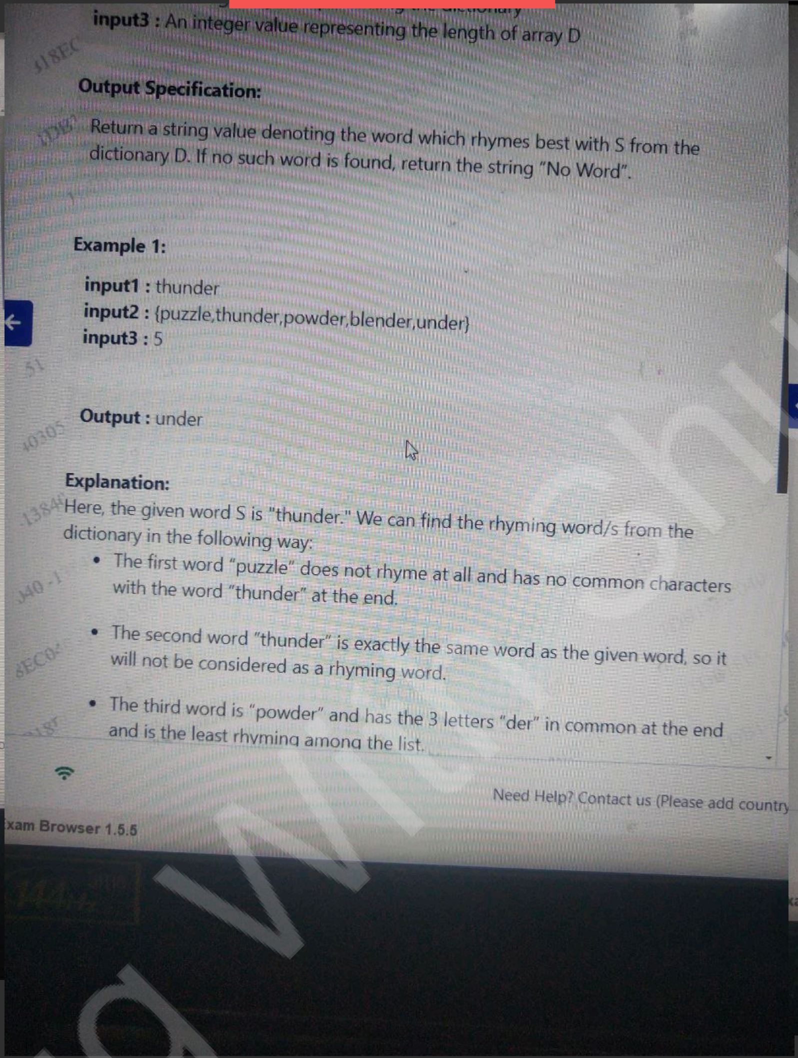
Your task is to find and return a string value denoting the word which rhymes best with S, from the dictionary D. If no such word is found, return the string "No Word".

Note:

- *If all the characters match, it is the same word and not a rhyming word.*
- *All the given words are in lowercase.*
- *If multiple rhyming words are found, then choose the word with the least index.*

Input Specification:

input1 : A string value S, representing a single word
input2 : A string array D, representing the dictionary
input3 : An integer value representing the length of array D



input3 : An integer value representing the length of array D

Output Specification:

Return a string value denoting the word which rhymes best with S from the dictionary D. If no such word is found, return the string "No Word".

Example 1:

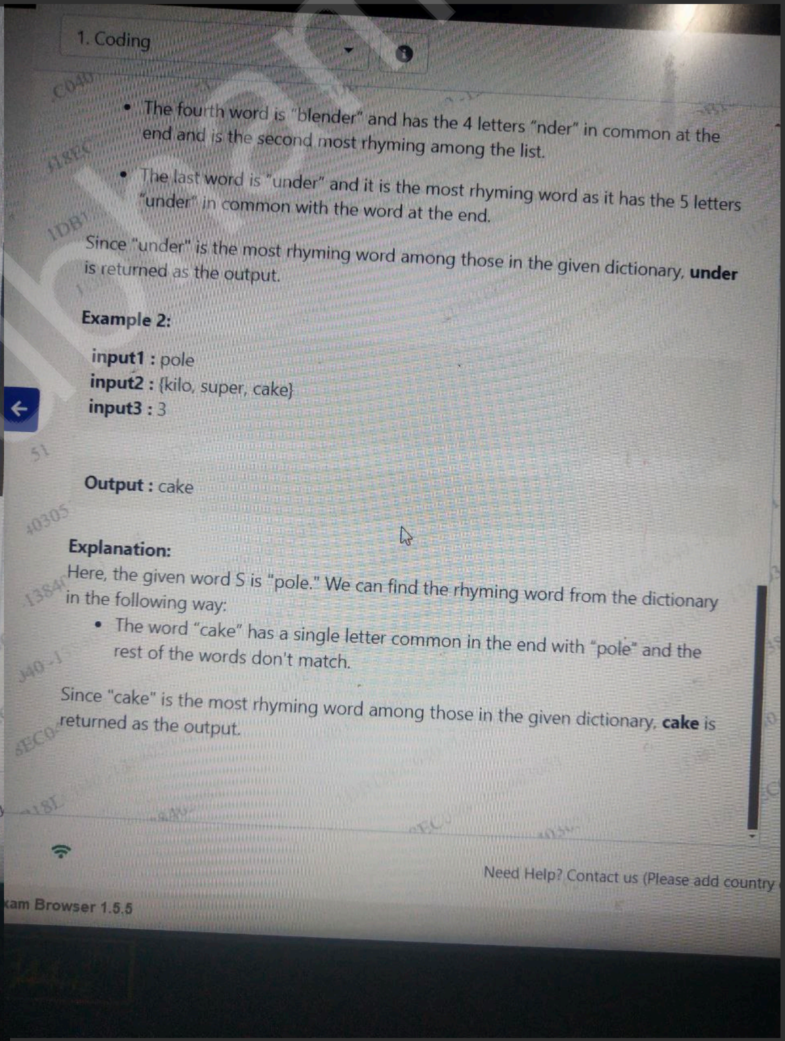
input1 : thunder
input2 : {puzzle,thunder,powder,blender,under}
input3 : 5

Output : under

Explanation:

Here, the given word S is "thunder." We can find the rhyming word/s from the dictionary in the following way:

- The first word "puzzle" does not rhyme at all and has no common characters with the word "thunder" at the end.
- The second word "thunder" is exactly the same word as the given word, so it will not be considered as a rhyming word.
- The third word is "powder" and has the 3 letters "der" in common at the end and is the least rhyming among the list.



- The fourth word is "blender" and has the 4 letters "nder" in common at the end and is the second most rhyming among the list.
- The last word is "under" and it is the most rhyming word as it has the 5 letters "under" in common with the word at the end.

Since "under" is the most rhyming word among those in the given dictionary, **under** is returned as the output.

Example 2:

input1 : pole
input2 : {kilo, super, cake}
input3 : 3

Output : cake

Explanation:

Here, the given word S is "pole." We can find the rhyming word from the dictionary in the following way:

- The word "cake" has a single letter common in the end with "pole" and the rest of the words don't match.

Since "cake" is the most rhyming word among those in the given dictionary, **cake** is returned as the output.

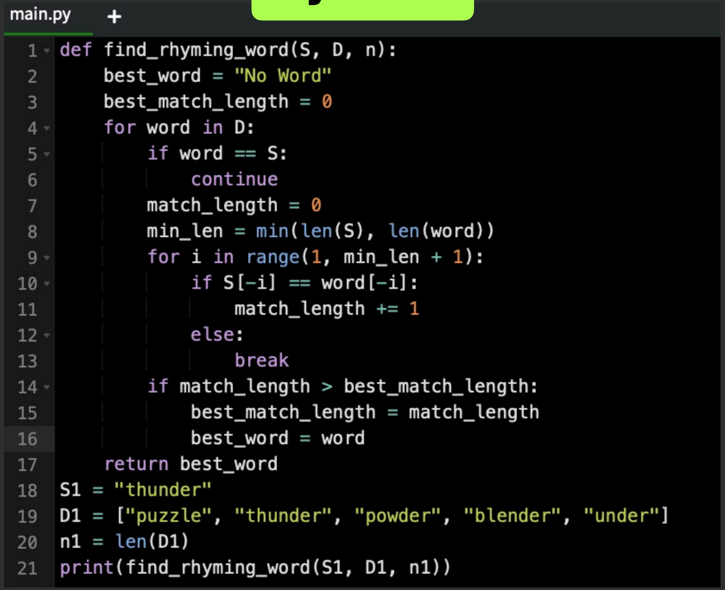
```
S1 = "thunder"
D1 = ["puzzle", "thunder", "powder", "blender", "under"]
n1 = len(D1)
```

Output: under

```
S2 = "pole"
D2 = ["kilo", "super", "cake"]
n2 = len(D2)
```

Output: cake

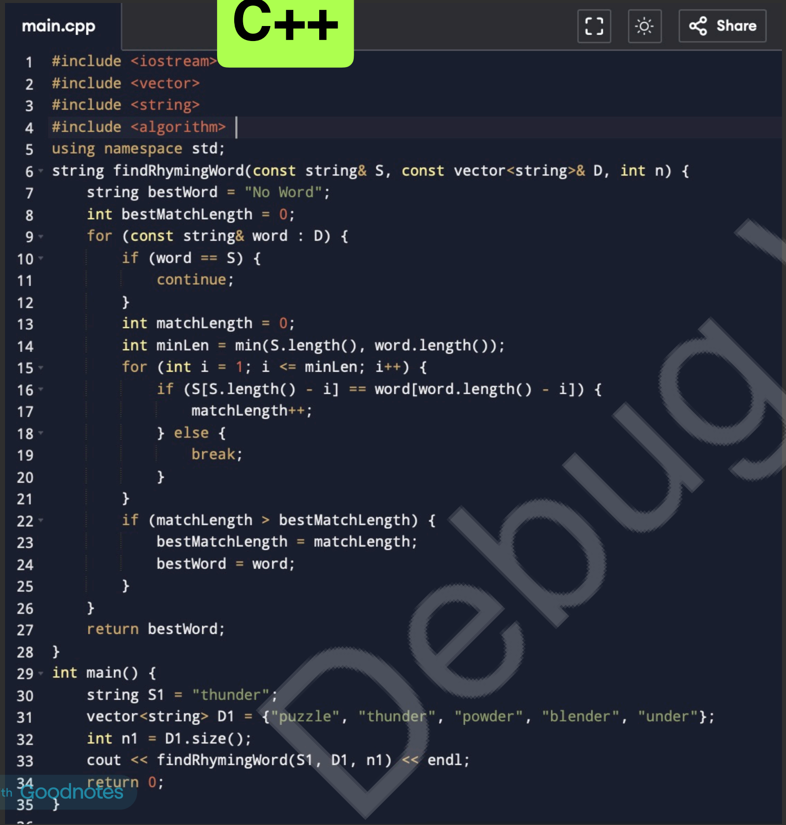
Python



main.py

```python
def find_rhyming_word(S, D, n):
    best_word = "No Word"
    best_match_length = 0
    for word in D:
        if word == S:
            continue
        match_length = 0
        min_len = min(len(S), len(word))
        for i in range(1, min_len + 1):
            if S[-i] == word[-i]:
                match_length += 1
            else:
                break
        if match_length > best_match_length:
            best_match_length = match_length
            best_word = word
    return best_word
S1 = "thunder"
D1 = ["puzzle", "thunder", "powder", "blender", "under"]
n1 = len(D1)
print(find_rhyming_word(S1, D1, n1))
```

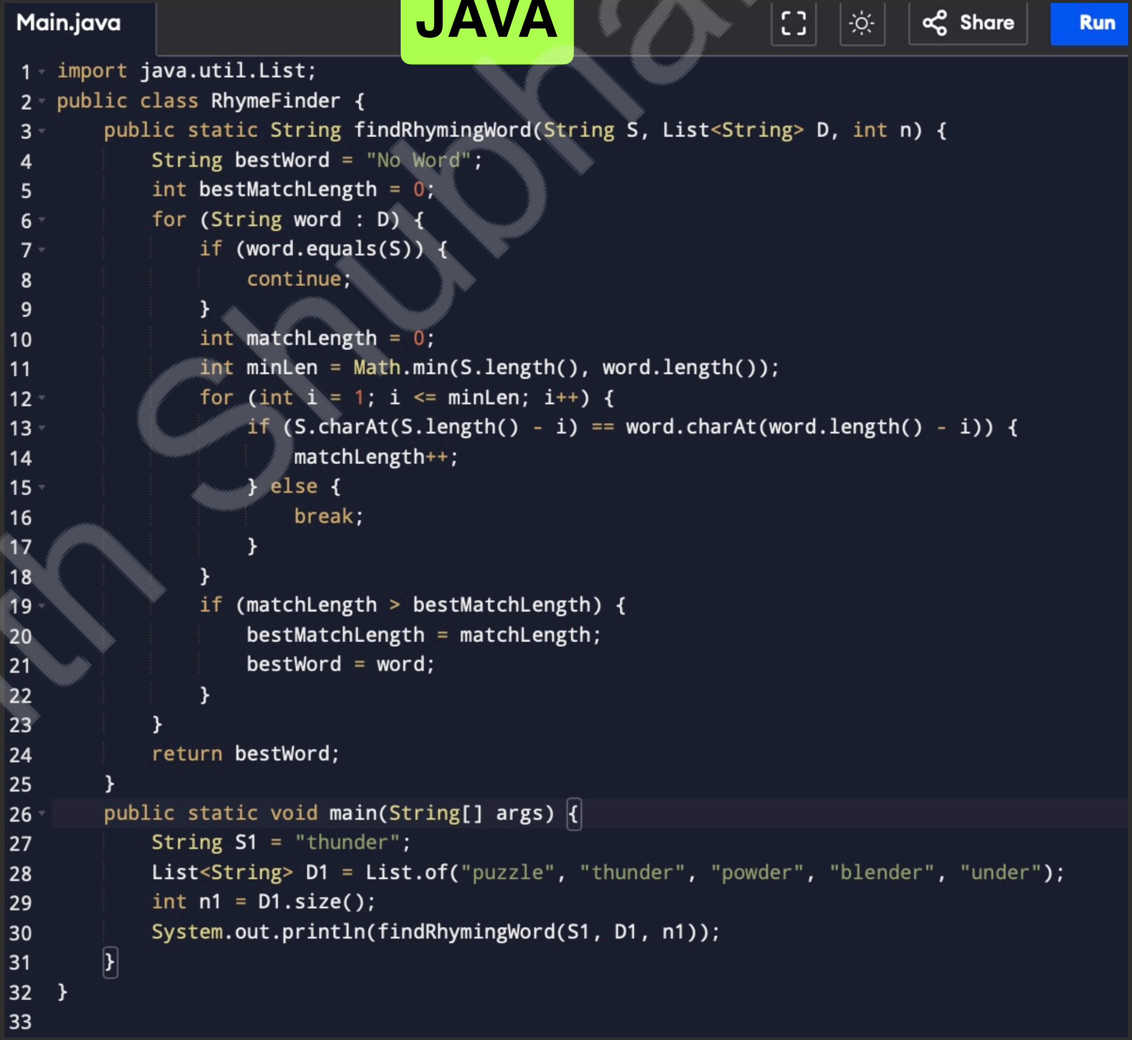
C++



main.cpp

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;
string findRhymingWord(const string& S, const vector<string>& D, int n) {
    string bestWord = "No Word";
    int bestMatchLength = 0;
    for (const string& word : D) {
        if (word == S) {
            continue;
        }
        int matchLength = 0;
        int minLen = min(S.length(), word.length());
        for (int i = 1; i <= minLen; i++) {
            if (S[S.length() - i] == word[word.length() - i]) {
                matchLength++;
            } else {
                break;
            }
        }
        if (matchLength > bestMatchLength) {
            bestMatchLength = matchLength;
            bestWord = word;
        }
    }
    return bestWord;
}
int main() {
    string S1 = "thunder";
    vector<string> D1 = {"puzzle", "thunder", "powder", "blender", "under"};
    int n1 = D1.size();
    cout << findRhymingWord(S1, D1, n1) << endl;
    return 0;
}
```

JAVA



Main.java

```java
import java.util.List;
public class RhymeFinder {
    public static String findRhymingWord(String S, List<String> D, int n) {
        String bestWord = "No Word";
        int bestMatchLength = 0;
        for (String word : D) {
            if (word.equals(S)) {
                continue;
            }
            int matchLength = 0;
            int minLen = Math.min(S.length(), word.length());
            for (int i = 1; i <= minLen; i++) {
                if (S.charAt(S.length() - i) == word.charAt(word.length() - i)) {
                    matchLength++;
                } else {
                    break;
                }
            }
            if (matchLength > bestMatchLength) {
                bestMatchLength = matchLength;
                bestWord = word;
            }
        }
        return bestWord;
    }
    public static void main(String[] args) {
        String S1 = "thunder";
        List<String> D1 = List.of("puzzle", "thunder", "powder", "blender", "under");
        int n1 = D1.size();
        System.out.println(findRhymingWord(S1, D1, n1));
    }
}

```