BETA Can't read the text? Switch theme

1. Cost of Path in a Directed Graph

In a directed graph, each edge of the graph is assigned a cost. This cost helps the programmers to identify the least cost path from a source vertex to a destination vertex.

Given a directed acyclic graph with the cost for each of edges, you are expected to write a program to find the cost of the path that is to be traversed between a given source and destination.

Consider the input where is source vertex and is destination vertex and value is the cost of path to traverse from to . Similarly, the other inputs are as below:

```
A, B, 2
B, C, 5
A, C, 9
C, D, 3
D, E, 2
```

Assume that path for which the cost is to be calculated, is given as: to to . This means the and thus the consi it works out to

Assume that path for which the cost is to be calculated, is given as: . This means the traversal path is to to and thus the considering the cost of each path then it works out to 12.

Function Description

Complete the function *computPathCost* in the editor below. The function must print the cost of the path.

computPathCost has the following parameter(s):

costOfEdges: a list of costs between 2 vertices in the format `sourceVertex,destinationVertex,cost`.

path: the path for which the cost is to be computed.

Should any of the constraints (mentioned below) is violated, then the output should be:

Constraints

- A minimum of 3 vertices should be provided in the input. There is no upper limit on this.
- The vertices would be named in uppercase alphabets.
- Should there an invalid vertex in the input *path*, then mark as invalid input.

▶ **Input Format For Custom Testing**

Input Format For Custom Testing

The first line contains an integer, *n*, denoting the number of elements in costOfEdges variable.

Each line *i* of the *n* subsequent lines (where $0 \leq i < n$) contains cost of each edge describing costOfEdges[i].

The last line is path for which the cost is to be calculated.

Sample Case 0

Sample Input For Custom Testing

```
5
A,B,2
B,C,5
A,C,9
C,D,3
D,E,2
A-C-D
```

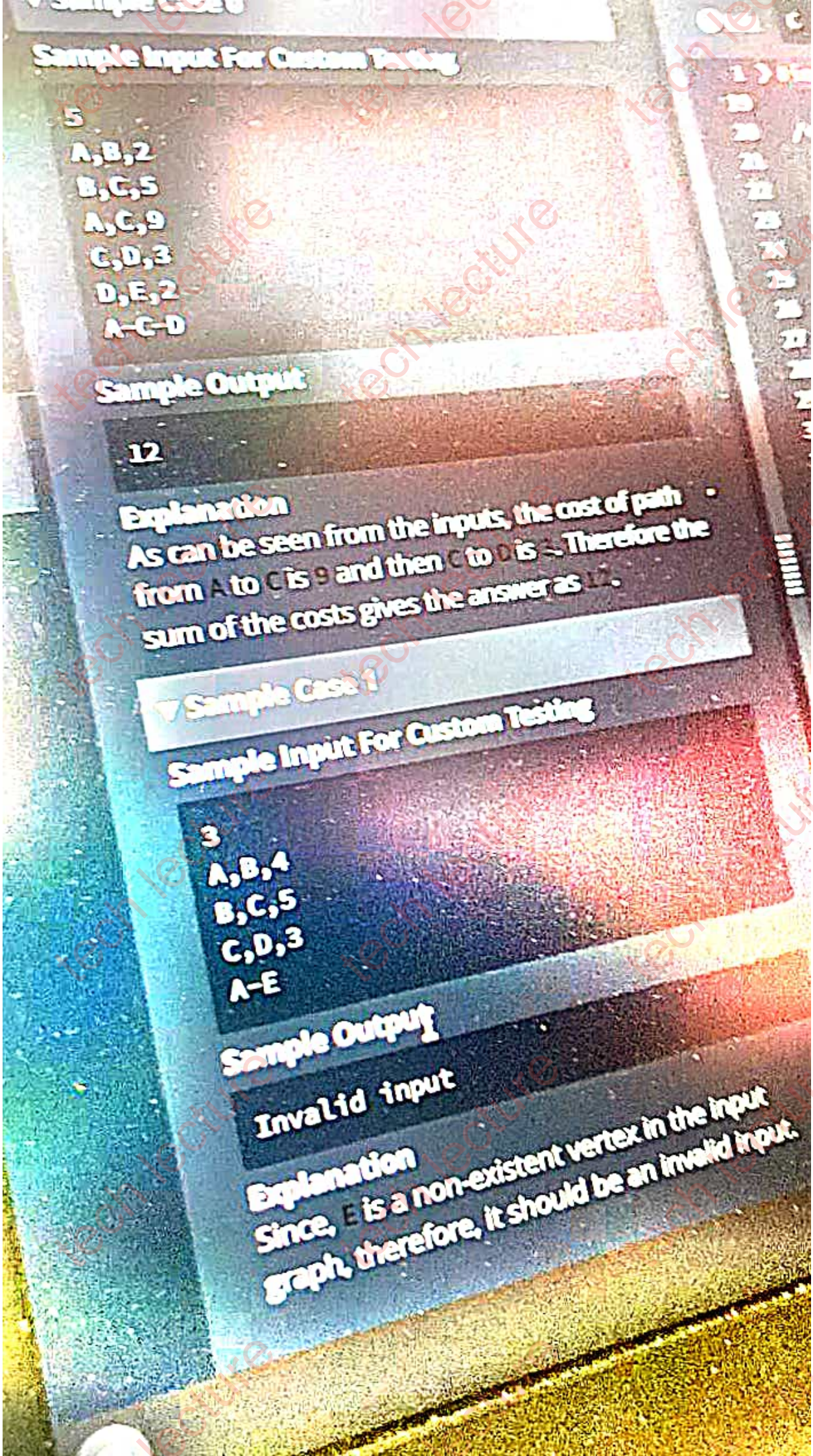
Sample Output

```
12
```

Explanation

As can be seen from the inputs, the cost of path from A to C is 9 and then C to D is 3. Therefore the sum of the costs gives the answer as 12.

Sample Case 1


Sample Input For Custom Testing
5
A,B,2
B,C,5
A,C,9
C,D,3
D,E,2
A-C-D
Sample Output
12
Explanation
As can be seen from the inputs, the cost of path
from A to C is 9 and then C to D is 3. Therefore the
sum of the costs gives the answer as 12.
Sample Case 1
Sample Input For Custom Testing
3
A,B,4
B,C,5
C,D,3
A-E
Sample Output
Invalid input
Explanation
Since, E is a non-existent vertex in the input
graph, therefore, it should be an invalid input.

29m left

BETA Can't read the text? Switch theme

ALL

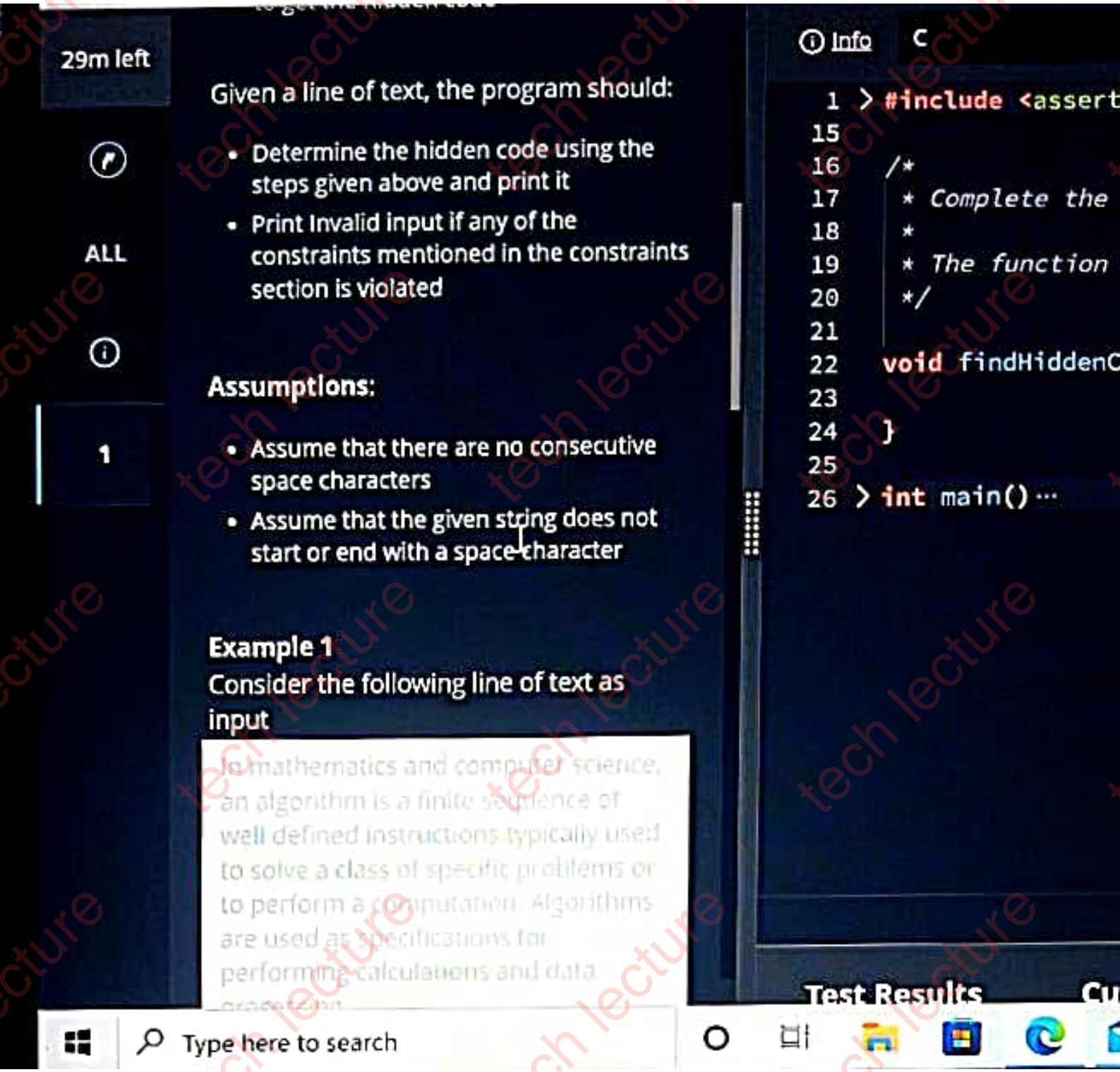
1

1. Find hidden code in text using factorization

In a particular text encoding algorithm, codes are hidden within plain text. The steps to extract the code is as follows:

- Find the length of the given text and factorize this length to determine its factors. For example, if the length is 10 the factors are 1, 2, 5, 10
- Use these factors as positions and find the characters in the text that come in that location. For example, in the case of 10, find the 1st, 2nd, 5th and 10th characters
- If the character at a particular position is a space character, pick the character next to it.
- Print these characters in the same order to get the hidden code

Given a line of text, the program should:


29m left
ALL
1
Given a line of text, the program should:
• Determine the hidden code using the steps given above and print it
• Print Invalid input if any of the constraints mentioned in the constraints section is violated
Assumptions:
• Assume that there are no consecutive space characters
• Assume that the given string does not start or end with a space character
Example 1
Consider the following line of text as input
In mathematics and computer science, an algorithm is a finite sequence of well defined instructions typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data
Info
C
1 > #include <assert
15
16 /*
17 * Complete the
18 *
19 * The function
20 */
21
22 void findHidden
23
24 }
25
26 > int main() ···
Test Results
Type here to search

29m left

ALL

1

The length of this text is: 268
Factors of 268: 1, 2, 4, 67, 134, 268
The hidden code is determined by finding the 1^{st}, 2^{nd}, 4^{th}, 67^{th}, 134^{th} and 268^{th} characters from this text: I n m e f .
The output is:

Inmef.

Example 2
Consider the following line of text as input

In mathematics and computer science, an algorithm is a finite sequence of well defined instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing

The length of this text is: 269
269 is a prime number, and this violates a constraint mentioned in the constraints section below.
The output is:

Type here to search


Status check page
29m left
ALL
1
▶ Input Format For Custom Testing
▼ Sample Case 0
Sample Input For Custom Testing
In mathematics and computer science, an algorithm is a finite sequence of well defined instructions typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing.
Sample Output
lnmef.
Explanation
Applying the given steps in description, the positions of the hidden characters in the text are: 1, 2, 4, 67, 134, 268. Hence the output.
▶ Sample Case 1
Info
C
1 > #include <assert
15
16 /*
17 * Complete the
18 *
19 * The function
20 */
21
22 void findHiddenC
23
24 }
25
26 > int main()
Test Results

11. Find number of empty fields in a record

A CSV or Comma Separated Value format is used to store records in the form of delimited text. In this a record is simply a line of text, with each field value separated using a delimiter character.
One of the most commonly delimiter character is the comma character. For example this is one line of text with comma as delimiter: 101,Bob,180,75. Here the field values are: **101 Bob 180 75**

The given record has blank fields if:

1. The record starts with a delimiter - this means the first field is empty
2. There are no characters between any pair of delimiters - if two delimiters occur adjacent to each other, it means the field in between is empty
3. The record ends with a delimiter - this means the last field is empty

Given a line of text, and the delimiter character and a default field value, the program should:

- Print the given line of text after replacing the empty fields with the given default value on the first line and number of such fields that were replaced in the second line
- Print Invalid input if any constraint mentioned in the constraint section below is violated

Given a line of text, and the delimiter character and a default field value, the program should:

- Print the given line of text after replacing the empty fields with the given default value on the first line and number of such fields that were replaced in the second line
- Print Invalid input if any constraint mentioned in the constraint section below is violated

Example 1

Consider the following input:

```
01,Bob,180,75
,
Null
```

Here the delimiter character is , and the default value is **Null**

Here, we can see that the record does not begin or end with the delimiter, and there are no consecutive delimiters occurring. Hence there are no missing fields and the output is:

```
01,Bob,180,75
0
```

Example 2

Consider the following input:

```
111::A::B:
:
Null
```

Here, we can see that the record does not begin or end with the delimiter, and there are no consecutive delimiters occurring. Hence there are no missing fields and the output is:

```
01,Abc,180,75
0
```

Example 2

Consider the following input:

```
111::A::B:
:
Null
```

Here the delimiter character is **:** and the default value is **Null**

Here, we can see that the record ends with the delimiter, which means the last field is missing. Also, there are two consecutive pair of delimiters occurring in between. Thus there are two fields missing in between. The total number of missing fields are 3. These have to be replaced with the default value. The output is:

```
111:Null:A:Null:B:Null
3
```

Function Description

Complete the function *replaceMissingFields* in the editor below.

Function Description

Complete the function *replaceMissingFields* in the editor below.

The function must:

- Print given line of text after replacing the empty fields with the given default value on the first line and number of such fields that were replaced in the second line
- Print Invalid Input if any constraint mentioned in the constraint section below is violated

replaceMissingFields has the following parameter(s):

record: a String

delimiter: a Character

defaultValue: a String

Constraints

- The record should have at least one valid field. That is, not all fields should be missing
- There should be more than 3 fields including empty ones

▶ Input Format For Custom Testing

▼ Sample Case 0

1. Extract user message from transcript

A particular chat program saves the chat transcript containing user names and user messages as follows:

- Each message will be in a new line
- Each message begins with the user name
- The user name is followed by a combination of a space, a colon character, and one more space :
- The user's message appears after combination mentioned above

Given a chat transcript in the above format, a username and a number n, the program should print the nth message by that particular user. In case any of the constraints mentioned below are violated, print Invalid input

Example 1

Consider the following chat transcript:

```
Alice : Good morning Bob
Bob : Good morning, Alice!
Alice : The weather seems nice today.
```

Consider the following additional inputs: `Alice, 2`
The program should print the **2nd** message by the user **Alice** from the transcript above. The output would be:

1. Cost of path in a directed graph (done)

main.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
    int s;
    cin>>s;
    if(s==5)
    {
        cout<<"12";
    }
    else
    {
        cout<<"Invalid input";
    }
}

```

2. Extract user message from transcript

main.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
    string s;
    cin>>s;
    if(s=="Alice")
    {
        cout<<"The weather seems nice today.";
    }
    else
    {
        cout<<"Invalid input";
    }
    return 0;
}

```

3. Find number of empty fields in a record

main.cpp

```c
#include <stdio.h>

int main()
{
  int blank_char, tab_char, new_line;
  blank_char = 0;
  tab_char = 0;
  new_line = 0;
  int c;
  printf("Number of blanks, tabs, and newlines:\n");
  printf("Input few words/tab/newlines\n");
  for (; (c = getchar()) != EOF;)
  {
    if ( c == ' ' ){
      ++blank_char;
    }
    if ( c == '\t' ){
      ++tab_char;
    }
    if ( c == '\n' ){
      ++new_line;
    }
  }
  printf("blank=%d,tab=%d,newline=%d\n",blank_char,tab_char,new_line);
}
```

4. Find hidden code in text using factorization

main.cpp

```cpp
#include <iostream>
using namespace std;
int main()
{
    string s;
    getline(cin,s);
    int n=s.length();
    string r="";
    for(int i=1; i<=n; i++){
        if(n%i==0)
        {
            if(s[i-1]==' ')
            r=r+s[i];
            else
            r=r+s[i-1];
        }
    }
    cout<<r;
    return 0;
}
```