1. We want to store the information about different vehicles. Create a class named Vehicle with two data member named mileage and price. Create its two subclasses
*Car with data members to store ownership cost, warranty (by years), seating capacity and fuel type (diesel or petrol).
*Bike with data members to store the number of cylinders, number of gears, cooling type(air, liquid or oil), wheel type(alloys or spokes) and fuel tank size(in inches)
Make another two subclasses Audi and Ford of Car, each having a data member to store the model type. Next, make two subclasses Bajaj and TVS, each having a data member to store the make-type.
Now, store and print the information of an Audi and a Ford car (i.e. model type, ownership cost, warranty, seating capacity, fuel type, mileage and price.) Do the same for a Bajaj and a TVS bike.
*************************************************************************************
2. A class Point is declared as follows:

```
class Point{
public:
Point(int=0;int=0);  //default constructor
void setPoint(int,int); //set coordinates
int getX() const {return x;} //get x coordinates
int getY() const {return y;} //get y coordinates
void printPoint(); // print (x,y) coordinates
private:
int x,y;
};
```

Write the implementation of the class Point in the same file.
Then, declare a class called Circle that is derived from the class Point. The class Circle has public member functions Circle (constructor), setRadius(), getRadius() and area() and one private data member radius. The class Circle indirectly uses private member x and y of class Point to store the coordinate of the center of the circle. The class Circle also checks to make sure the radius value is a positive number, otherwise it is set to default value 1.
Note: The private members of class Point can only be indirectly accessed by class Circle using public methods of class point.
Implement the class Circle and write a driver program to test the class Circle. An example of the output of the driver program is.

```
Enter x: 2
Enter y: 2
Enter radius: 1
Circle center is (2,2)
Radius is 1
Area is 3.14
```

*************************************************************************************
3. Make a base class named Item then define its private and public members. Define a constructor with parameters for both data members (Barcode, Item Name). You should add default values for the parameters to provide a default constructor for the class. In addition to the access methods setCode() and getCode(), you are also required to define the methods scanner() and printer(). These methods will simply output item data on screen or read the data of an item from the keyboard.Define two derived classes PackedFoodand FreshFood. In addition to the Item class data, the PackedFood class should contain the unit price. The FreshFood class should contain a weight and a price per kilogram as data members. You are required to define a constructor with parameters providing default-values for all data members in both classes. Also define the access methods needed for the new data members.
Make the main() function then test the classes in it, that creates two objects each of the types Item, PackedFood and FreshFood. One object of each type should be fully initialized in the object definition. You can use the default constructor to create the other object.
The get() and set() methods and the scanner() method should be well written and the printer() method should display the items on screen.
*************************************************************************************
4. Create an application for reading books using files. It has two interfaces 1. User 2. Admin. Take 3 books as a separate text file. If the user want to read a book, then he/she needs to register and the registration needs their name, userId, emailId, mob no. Once the registration is completed, you need to send a access request to the admin to read a book and he may approve by giving a password, then the user can enter the password and read the book or he may reject his/her approval.
*************************************************************************************
5. Word Ladder
You are given two words - beginWord and endWord. You also have a wordList of size n. All words are of the same length.

You have to create a transformation sequence from beginWord to endWord that looks like this:
beginWord → str1 → str2 → str3 → ... → strk

Adjacent pairs in the sequence differ by a single character, and all strings from str1 to strk lie in wordList.
Also strk = endWord.

Find the minimum number of words in the shortest possible transformation sequence. If such a sequence is not possible return 0.

Example:
beginWord = "work"
endWord = "tech"
wordList = ["toch","worh","wock","woch","tech","werh"]
Output: 5
Explanation: Shortest transformation sequence is "work" → "wock" → "woch" → "toch" → tech", which is 5 words long.

Input Format
The first line contains an integer T denoting the number of test cases.

For each test case, the input has three lines:

Two space-separated strings beginWord and endWord.
An integer n denoting the number of words in the wordList.
n space-separated strings denoting the word list.

Output Format
For each test case, the output has a line containing the shortest possible transformation sequence.

Sample Input
2
work tech
6
toch worh wock woch tech werh

black white
7
aracz blacz bracz ahacz white ahace ahice

Expected Output
5
0

1 <= T <= 10
1 <= n <= 5000
1 <= string length <= 5
*************************************************************************************
6. Boggle Board
M S E F
R A T D
L O N E
K A F B ===>4X4 Boggle Board
Generate a list of possible words from a character matrix.
Given an M × N boggle board, find a list of all possible words that can be formed by a sequence of adjacent characters on the board.We are allowed to search a word in all eight possible directions, i.e.,
North, West, South, East, North-East, North-West, South-East, South-West,
but a word should not have multiple instances of the same cell.

Consider the following the traditional 4 × 4 boggle board.
If the input dictionary is [START, NOTE, SAND, STONED], the valid words are [NOTE, SAND, STONED].

Practice This Problem
We can use Depth–first search (DFS) to solve this problem. The idea is to start from each character in the matrix and explore all eight paths possible and recursively check if they lead to a solution or not. To make sure that a word doesn't have multiple instances of the same cell, keep track of cells involved in the current path in the matrix, and before exploring any cell, ignore the cell if it already covered in the current path.
*************************************************************************************