# Sitecore Product Support pre-interview tasks

## Introduction

## We are pleased to provide you with the following tasks as part of the interview process for the Sitecore Product Support position. Complete as many tasks as you can up to a level of complexity that you are confident with. Recommendation is not to invest more than 4 hours on all tasks.

## Task #1. Simple figures

**Difficulty 1:**

Your task is to design a collection of classes representing simple geometric figures, which include Point, Line, and Circle.  
  
**Difficulty 2:**

Includes Difficulty 1 and the following:

Each class representing a geometric figure must provide methods for Move and Rotate actions. Full method implementation is required for each geometric figure.  
  
**Difficulty 3:**

Includes Difficulty 2 and the following:

Create another class named "Aggregation" that can hold a varying number of various geometric figure instances (e.g., 2 Points and a Circle, or 5 Lines and a Point). When an Aggregation instance is moved or rotated, all its contained figures should react in the same manner. Please provide a full implementation of the Aggregation class, including all methods and class definition.

## Task #2. Palindrome

A palindrome is a sequence of characters that retains its identity when read in reverse, such as madam, racecar, or the number 10201.  
  
**Difficulty 1:**

Your task is to create a method that takes in two strings: an InputString and a TrashSymbolString. This method should verify if the InputString is a palindrome, disregarding all symbols found in the TrashSymbolString and ignoring the case.  
  
For instance:  
  
Example 1:  
InputString: "a@b!!b$a"  
TrashSymbolsString: "!@$"  
Expected result: true  
  
Example 2:  
InputString: "?Aa#c"  
TrashSymbolsString: "#?"  
Expected result: false  
  
**Difficulty 2:**

Includes Difficulty 1 and the following:

In your implementation, ensure that no TrashSymbols are removed from the InputString and that no new strings are created.  
  
**Difficulty 3:**

Includes Difficulty 2 and the following:

Your method should only scan the InputString once.

## Task #3. Minefield

**Difficulty 1:**  
Welcome to a minefield of n x m dimensions, where random fields are booby-trapped with a bomb. Rest assured, a safe path through this minefield always exists. We have a special dog, named "Sniffer Pup", who has:

1. ability to detect if any of the eight fields adjacent to him (up, down, left, right, and the four diagonals) are rigged with a bomb.
2. ability step to any of the eight fields adjacent to him (up, down, left, right, and the four diagonals)

Your challenge is to create an algorithm that guides Sniffer Pup through the minefield safely, using his remarkable bomb-detection skill.

Example   
√ - safe path  
X – bomb

ヽ(°ᴥ°)ﾉ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | √ | X | X |  |
| X | X | √ | X |  |
|  | X | X | √ | X |
| X |  | X | √ | X |
|  | X | √ | X | X |

**Difficulty 2:**

Includes Difficulty 1 and the following:

A girl, Ally, is following Sniffer Pup's trail. Ally always steps onto the field that Sniffer Pup just vacated. However, it's important to remember that Sniffer Pup and Ally cannot occupy the same field at the same time. Your challenge is to design an algorithm that ensures the safe passage for both Sniffer Pup and Ally through the minefield, considering Sniffer Pup's bomb detection ability and Ally's path.

**Difficulty 3:**

Includes Difficulty 2 and the following:

Please provide a C# code implementation that accomplishes the tasks described above.