```
using System;
using System.Collections.Generic;
using System.Linq;

namespace MyConsoleCSElisa
{
    class Program
    {

        private static List<KeyWord> Keywords = new List<KeyWord> {
            new KeyWord { Category = 1, Word = "Bruder" }, new KeyWord { Category = 1, Word = "Vater" }, new KeyWord { Category
                   = 1, Word = "Sohn" }, new KeyWord { Category = 1, Word = "Opa" }, new KeyWord { Category = 1, Word =
                   "Freund" },
            new KeyWord { Category = 2, Word = "Mutter" }, new KeyWord { Category = 2, Word = "Schwester" }, new KeyWord {
                   Category = 2, Word = "Tochter" }, new KeyWord { Category = 2, Word = "Oma" },
            new KeyWord { Category = 3, Word = "Gewalt" }, new KeyWord { Category = 3, Word = "Druck" }, new KeyWord { Category
                   = 3, Word = "Schweigen" } };

        private static List<Answer> Answers = new List<Answer> {
        /*  0 */ new Answer { Counter = 0, AnswerSentence = "Dein {0} ist Dir sehr wichtig, nicht war?" },
        /*  1 */ new Answer { Counter = 0, AnswerSentence = "Haettest Du darueber nicht mit Deinem {0} sprechen sollen?" },
        /*  2 */ new Answer { Counter = 0, AnswerSentence = "Erzaehle mir mehr über die Beziehung zu Deinem {0}!" },
        /*  3 */ new Answer { Counter = 0, AnswerSentence = "Deine {0} ist Dir sehr wichtig, nicht wahr?" },
        /*  4 */ new Answer { Counter = 0, AnswerSentence = "Haettest Du darueber nicht mit Deiner {0} sprechen sollen?" },
        /*  5 */ new Answer { Counter = 0, AnswerSentence = "Erzaehle mir mehr ueber die Beziehung zu Deiner {0}!" },
        /*  6 */ new Answer { Counter = 0, AnswerSentence = "{0} ist keine echte Loesung." },
        /*  7 */ new Answer { Counter = 0, AnswerSentence = "Es ist nicht gut, mit {0} zu leben." },
        /*  8 */ new Answer { Counter = 0, AnswerSentence = "Sollte die Welt nicht auf {0} verzichten?" },
        /*  9 */ new Answer { Counter = 0, AnswerSentence = "Was bedeutet das eigentlich fuer Dich: {0}?" } };

        private static List<Binding> Bindings = new List<Binding> {
            new Binding { KeyWordCategory = 1, AnswersId = 0 }, new Binding { KeyWordCategory = 1, AnswersId = 1 }, new Binding
              { KeyWordCategory = 1, AnswersId = 2 },
            new Binding { KeyWordCategory = 2, AnswersId = 3 }, new Binding { KeyWordCategory = 2, AnswersId = 4 }, new Binding
              { KeyWordCategory = 2, AnswersId = 5 },
            new Binding { KeyWordCategory = 3, AnswersId = 6 }, new Binding { KeyWordCategory = 3, AnswersId = 7 }, new Binding
              { KeyWordCategory = 3, AnswersId = 8 },
            new Binding { KeyWordCategory = 1, AnswersId = 9 }, new Binding { KeyWordCategory = 2, AnswersId = 9 }, new Binding
              { KeyWordCategory = 3, AnswersId = 9 } };

        private static List<Phrase> Phrases = new List<Phrase> {
            new Phrase{Counter = 0, PhraseSentence = "Ich verstehe Deine Zurueckhaltung."},
            new Phrase{Counter = 0, PhraseSentence = "Soltest Du nicht offener von Dir reden?"},
            new Phrase{Counter = 0, PhraseSentence = "Was meinst Du, ist denn die Ursache von all dem?"},
            new Phrase{Counter = 0, PhraseSentence = "Kannst Du etwas paeziser werden?"},
            new Phrase{Counter = 0, PhraseSentence = "Du soltest nicht alles in Dich hineinfressen."},
            new Phrase{Counter = 0, PhraseSentence = "Fuehlst Du Dich in dieser Hinsicht unsicher?"} };

        private static Random _random = new Random();
        private static int _lastPhrase = -1;

        /// <summary>
        /// RandomDouble
        /// </summary>
        /// <param name="min"></param>
        /// <param name="max"></param>
        /// <param name="deci"></param>
        /// <returns></returns>
        private static double RandomDouble(double min, double max, int deci)
        {
            double d;
            d = _random.NextDouble() * (max - min) + min;
            return Math.Round(d, deci);
        }

        /// <summary>
        /// SegmentWordsLinq
        /// </summary>
        /// <param name="statement"></param>
        /// <returns></returns>
        private static List<string> SegmentWordsLinq(string statement)
        {
            List<string> words = new List<string>();

            words = statement.Split(' ').ToList();

            return words;
        }

        /// <summary>
        /// SegmentWordsSplit
        /// </summary>
        /// <param name="statement"></param>
        /// <returns></returns>
        private static List<string> SegmentWordsSplit(string statement)
        {
            List<string> words = new List<string>();
    string[] tokens = statement.Split(' ');
    for (int i = 0; i < tokens.Length; i++)
        words.Add(tokens[i]);

    return words;
}

/// <summary>
/// DunpWordList
/// </summary>
/// <param name="words"></param>
private static void DumpWordList(List<string> words)
{
    Console.WriteLine(String.Format("STATEMENT:"));
    foreach (var w in words)
        Console.Write(String.Format("{0} ", w));

    Console.WriteLine(String.Format("\nWORD LIST:"));
    foreach (var w in words)
        Console.WriteLine(String.Format("{0}", w));
}

/// <summary>
/// SearchForAnswer
/// </summary>
/// <param name="words"></param>
/// <returns></returns>
private static string SearchForAnswer(List<string> words)
{
    string answer;
    int answerId;

    foreach(var w in words)
        foreach(var k in Keywords)
            if(w == k.Word)
            {
                int id = Keywords.IndexOf(k);
                answerId = GetAnswerId(k.Category);
                if(answerId >= 0)
                {
                    answer = Answers[answerId].AnswerSentence;
                    answer = String.Format(answer,w);
                    Answers[answerId].Counter += 10;
                    return answer;
                }
            }

    // We have no keyword
    answerId = GetPhraseId();
    answer = Phrases[answerId].PhraseSentence;
    Phrases[answerId].Counter += 10;
    return answer;
}

/// <summary>
/// GetPhraseId
/// </summary>
/// <returns></returns>
private static int GetPhraseId()
{
    int id = _lastPhrase;

    // repeat as long as it's not the same phrase as before
    while (id == _lastPhrase)
        id = (int)RandomDouble(0, Phrases.Count - 1, 0);
    _lastPhrase = id;

    Phrases[id].Counter++;

    return id;
}

/// <summary>
/// GetAnswerId
/// </summary>
/// <param name="id"></param>
/// <returns></returns>
private static int GetAnswerId(int id)
{
    int min = Int32.MaxValue;
    int minID = -1;
    int answerId = -1;

    foreach (var b in Bindings)
        if (id == b.KeyWordCategory)
        {
```

```
                    answerId = b.AnswersId;
                    if(min > Answers[answerId].Counter)
                    {
                        minID = answerId;
                        min = Answers[answerId].Counter;
                        --Answers[answerId].Counter;
                    }
                }

            return minID;
        }

        /// <summary>
        /// Main
        /// </summary>
        /// <param name="args"></param>
        static void Main(string[] args)
        {
            Console.WriteLine("Program MyConsoleCSElisa type [Ctrl] [C] to Exit and /? on command line for help");
            Console.WriteLine(String.Format("uhwgmxorg Version {0}\n",
               System.Reflection.Assembly.GetExecutingAssembly().GetName().Version));

            string statement;
            string answer;
            List<string> words;

            ProcessCommandLineOptions(args);

            Console.WriteLine("Sag was :-)\n");

            while (true)
            {
                statement = Console.ReadLine();
                words = SegmentWordsSplit(statement);
                answer = SearchForAnswer(words);
                Console.WriteLine(answer);
            }
        }

        /// <summary>
        /// ProcessCommandLineOptions
        /// </summary>
        /// <param name="cmdLine"></param>
        private static void ProcessCommandLineOptions(string[] cmdLine)
        {
            foreach (var c in cmdLine)
            {
                if (c.Contains("/?"))
                {
                    Console.WriteLine("Use:");
                    Console.WriteLine("  MyConsoleCSElisa /? for help");
                    Console.WriteLine("  MyConsoleCSElisa /k for show keywords");
                    Environment.Exit(1);
                }
                if (c.Contains("/k"))
                {
                    Console.WriteLine("You can use:");
                    foreach (var k in Keywords)
                        Console.Write(String.Format("{0} | ", k.Word));
                    Console.WriteLine("\nas keywords in a question/answer to Elisa\n");
                }
            }
        }
    }

    public class KeyWord
    {
        public int Category { get; set; }
        public string Word { get; set; }
    }
    public class Answer
    {
        public int Counter { get; set; }
        public string AnswerSentence { get; set; }
    }
    public class Binding
    {
        public int KeyWordCategory { get; set; }
        public int AnswersId { get; set; }
    }
    public class Phrase
    {
        public int Counter { get; set; }
        public string PhraseSentence { get; set; }
    }
}
```