```
/*
 * Created by SharpDevelop.
 * User: Davide
 * Date: 26/08/2012
 * Time: 18:19
 * 
 * To change this template use Tools | Options | Coding | Edit Standard Headers.
 */
using System;
using System.Collections.Generic;
using SMath.Manager;
using SMath.Math;

namespace combinFuncton
{
	/// <summary>
	/// SMath "combin" primitive
	/// </summary>
	public class MyClass : IPluginHandleEvaluation, IPluginLowLevelEvaluation
	{
		TermInfo[] termInfos;
		AssemblyInfo[] assemblyInfos;
		
		#region IPluginHandleEvaluation
		TermInfo[] IPluginHandleEvaluation.TermsHandled {
			get {
				return this.termInfos;
			}
		}
		
		AssemblyInfo[] IPlugin.Dependences {
			get {
				return this.assemblyInfos;
			}
		}
		
		void IDisposable.Dispose()
		{
			// Do nothing
		}
		
		void IPlugin.Initialize()
		{
			this.termInfos = new TermInfo[] {
				new TermInfo("combin", TermType.Function, "(n,k) - Returns the number of subsets (combinations) of k elements that can be 
formed from n elements.", FunctionSections.Unknown, true)
			};
			this.assemblyInfos = new AssemblyInfo[] {
				new AssemblyInfo("SMath Studio", new Version(0, 95), new Guid("a37cba83-b69c-4c71-9992-55ff666763bd"))
			};
```

```
        }
        #endregion

        #region IPluginLowLevelEvaluation
        bool IPluginLowLevelEvaluation.ExpressionEvaluation(Term root, Term[][] args, ref Store context, ref Term[] result)
        {
            if (root.Type == TermType.Function && root.Text == "combin" && root.ChildCount == 2){
                Term[]
                    arg1 = Decision.Preprocessing(args[0], ref context),
                    arg2 = Decision.Preprocessing(args[1], ref context);

                List<Term>
                    answer = new List<Term>();

                answer.AddRange(arg1);
                answer.Add(new Term(Operators.Factorial, TermType.Operator, 1));
                answer.AddRange(arg2);
                answer.Add(new Term(Operators.Factorial, TermType.Operator, 1));
                answer.AddRange(arg1);
                answer.AddRange(arg2);
                answer.Add(new Term(Operators.Substraction, TermType.Operator, 2));
                answer.Add(new Term(Operators.Factorial, TermType.Operator, 1));
                answer.Add(new Term(Operators.Multiplication, TermType.Operator, 2));
                answer.Add(new Term(Operators.Division, TermType.Operator, 2));

                result = answer.ToArray();

                return true;
            }
            return false;
        }
        #endregion
    }
}
```