```
1  //################### Compiler Design (2) ####################
2  //###################### Type Checking 1 #####################
3  //########## Input: expression var = val, var = var ##########
4  //################## Output: Matching result ##################
5  //################ Author: Ahmed A. Alsamman #################
6  using System;
7  using System.Collections.Generic;
8
9  namespace TypeCheckingVar
10 {
11
12     public struct Exp
13     {
14         public string lhs;
15         public string rhs;
16
17         public Exp(string var, string rhs)
18         {
19             this.lhs = var;
20             this.rhs = rhs;
21         }
22     }
23
24     public struct Token
25     {
26         public string name;
27         public string type;
28         public Token(string name, string type)
29         {
30             this.name = name;
31             this.type = type;
32         }
33     }
34
35     public class SymbolTable
36     {
37         public List<Token> token = new List<Token>();
38         public int Lookup(string tok)
39         {
40             for (int i = 0; i < token.Count; i++)
41                 if (token[i].name == tok)
42                     return i;
43             return -1;
44         }
45
46         public void Insert(string name, string type)
47         {
48             if (Lookup(name) == -1) // variable not found
49                 token.Add(new Token(name, type));
50         }
51     }
52
53
54
55
56
```

```
57
58      class Program
59      {
60          static string LHSType = null;
61          static string RHSType = null;
62          static SymbolTable symbTab = new SymbolTable();
63
64          static void Main(string[] args)
65          {
66
67              symbTab.Insert("a", "int");
68              symbTab.Insert("b", "double");
69
70              while (true)
71              {
72
73                  Console.Write("LHS: ");
74                  string lhs = Console.ReadLine();
75                  Console.Write("RHS: ");
76                  string rhs = Console.ReadLine();
77
78                  if (lhs == "" || rhs == "")
79                      continue;
80
81                  Console.WriteLine("Expression: {0}={1}", lhs, rhs);
82
83                  Exp exp = new Exp(lhs, rhs);
84
85                  if (Parse(exp) == "parsed")
86                      MatchTypes(exp);
87              }
88          }
89
90
91
92          private static bool validID(string var)
93          {
94              if (var[0] == '_' || Char.IsLetter(var[0]))
95              {
96                  for (int i = 1; i < var.Length; i++)
97                      if (!Char.IsLetter(var[i]) && !Char.IsDigit(var[i]) &&
                        var[i] != '_')
98                          return false;
99              }
100             else
101                 return false;
102
103             return true;
104         }
105
106
107
108
109
110
111
```

```
112
113         private static string Parse(Exp exp)
114          {
115            if (validID(exp.lhs))
116              symbTab.Insert(exp.lhs, null);
117            else
118            {
119             Console.WriteLine("Error: Expression must begin with a variable");
120              return null;
121            }
122            try
123            {
124              int.Parse(exp.rhs);
125              symbTab.Insert(exp.rhs, "int");
126            }
127              catch
128              {
129                  try
130                  {
131                      double.Parse(exp.rhs);
132                      symbTab.Insert(exp.rhs, "double");
133                  }
134                  catch
135                  {
136                      if (validID(exp.rhs))
137                          symbTab.Insert(exp.rhs, null);
138                      else
139                      {
140                          Console.WriteLine("Error: Invalid expression");
141                          return null;
142                      }
143                  }
144              }
145              return "parsed";
146          }
147         private static void MatchTypes(Exp exp)
148         {
149            LHSType = symbTab.token[symbTab.Lookup(exp.lhs)].type;
150            RHSType = symbTab.token[symbTab.Lookup(exp.rhs)].type;
151         if (LHSType == RHSType && LHSType != null)
152                Console.WriteLine("Type Mached");
153         else if (LHSType == null || RHSType == null)
154            {
155              if (LHSType == null)
156                 Console.WriteLine("Undefined variable {0}", exp.lhs);
157              if (RHSType == null )
158                 Console.WriteLine("Undefined variable {0}", exp.rhs);
159            }
160         else Console.WriteLine("Type Mismach. Expected a value of type \"{0}    ↵
              \"", LHSType);
161         }
162     }
163 }
164
```