Bug Summary

File:	Puzzle.cpp
Warning:	line 233, column 4
	Value stored to 'j' is never read

Report Bug

Annotated Source Code

```cpp
1   //Sudoku puzzle solver class
2   //Reference: http://web.cs.ucla.edu/~tianyi.zhang/tutorial.html
3
4   #include <iostream>
5   #include <vector>
6   #include <fstream>
7   #include <string>
8   #include <cstdlib>
9   #include "Puzzle.h"
10
11  using namespace std;
12
13  //Constructor
14  Puzzle::Puzzle(char* filename)
15  {
16      //declare object and open file
17      ifstream inSudokuFile(filename);
18      // If it did not open
19      if(!inSudokuFile)
20      {
21          cerr << "File could not be oppened" << endl;
22          exit(1);
23      }
24
25      int number;
26      vector<int> row;
27      //read in file
28      while(inSudokuFile>>number)
29      {
30          if(number >= 0 && number < 10) //if it's a number
31          {
32              row.push_back(number);//put number in a row
33              if(row.size() == 9)
34              {
35                  board.push_back(row);//push whole row
36                  row.erase(row.begin(),row.end());
37              }
38              if(board.size() == 9) break;//break once filled
39          }
40      }
41      //close file
42      inSudokuFile.close();
43
44      vector< vector<int> > grid;
45
46      //initialize possible values
47      int i,j,k;
48      for(i = 0;i < 9;i++)
49      {
```

```
50          for(j = 0;j < 9;j++)
51          {
52              for(k = 0;k < 9;k++)
53              {
54                  row.push_back(1);//set 1 is possible
55              }
56              grid.push_back(row);//push a row
57          }
58          possible.push_back(grid);//push whole gird
59      }
60  }
61
62  void Puzzle::solve()
63  {
64
65
66      int i,j,k,l,m,numpossible,newval,z,col,row;
67
68      //Set filled values to all 2's in possible
69      for(i = 0;i < 9;i++)//i is row
70      {
71          for(j = 0;j < 9;j++)//j is column
72          {
73              if((board[i][j]) != 0)//if there is already a value set possible to 2
74              {
75                  for(k = 0;k < 9;k++)//k is value
76                  {
77                      possible[i][j][k] = 2;//two means cell is filled
78                  }
79              }
80          }
81      }
82
83  //Loop and fill values
84  while(!done())//loop until solved
85  {
86      updatePossible();
87
88      //find and fill location that have only one possible value
89      for(i = 0;i < 9;i++)//i is row
90      {
91          for(j = 0;j < 9;j++)//j is column
92          {
93              if(possible[i][j][0] == 2) continue;//if full skip value
94              numpossible = 0;
95              for(k = 0;k < 9;k++)//k+1 is value
96              {
97                  if(possible[i][j][k] == 1)
98                  {
99                      numpossible++;
100                     newval = k+1;//if a number is possible store its value in newval
101                 }
102                 if(numpossible > 1) break;
103             }
104             if(numpossible == 1)
105             {
106                 fill(newval,i,j);//call fill to insert only possible value
107             }
108         }
109     }
110
111
```

```cpp
112        //seach columns for only one possible spot for a particular number
113        for(i = 0;i < 9;i++)//i is row
114        {
115            for(k = 0;k < 9;k++)//k+1 is value
116            {
117                numpossible=0;
118                for(j = 0;j < 9;j++)//j is column
119                {
120                    if(board[i][j] == k+1) break;//value in column already skip to next value
121                    if(possible[i][j][k] == 1)
122                    {
123                        numpossible++;
124                        col = j;
125                    }
126                    if(numpossible >1) break;//call fill to insert value
127                }
128                if(numpossible ==1)
129                {
130                    //cout<<"column search: ";
131                    fill(k+1,i,col);
132                }
133            }
134        }
135
136        //search rows for only one possible spot for a particular number
137        for(j = 0;j < 9;j++)//j is column
138        {
139            for(k = 0;k < 9;k++)//k+1 is value
140            {
141                numpossible=0;
142                for(i = 0;i < 9;i++)//i is row
143                {
144                    if(board[i][j] == k+1) break;//value in column already skip to next value
145                    if(possible[i][j][k] == 1)
146                    {
147                        numpossible++;
148                        row=i;
149                    }
150                    if(numpossible > 1) break;
151                }
152                if(numpossible == 1) //if only one possible position
153                {
154                    //cout<<"for column search: \n";
155                    fill(k+1,row,j);//call fill to insert value
156                }
157            }
158        }
159
160        //search blocks only one possible spot for a particular number
161        for(i = 0;i < 9;i+=3)//i is row
162        {
163            for(j = 0;j < 9;j+=3)//j is column
164            {
165                for(k = 0;k < 9;k++)//k+1 is value
166                {
167                    numpossible = 0;
168                    for(l = 0;l < 3;l++)//l is row
169                    {
170                        for(m = 0;m < 3;m++)//m is column
171                        {
172                            //cout<<"checking location:"<<i+l<<","<<j+m<<" for a "<<k+1<<endl;
173                            if(board[i+l][j+m] == k+1) break;//value in block already skip to next value
```

```cpp
                        if(possible[i+l][j+m][k] == 1)
                        {
                            numpossible++;
                            row = i+l;
                            col = j+m;
                            //cout<<"Numpossible incremeneted for "<<k+1<<" at "<<row<<","<<col<<endl;
                        }
                        if(numpossible > 1) break;
                    }
                    if(board[i+l][j+m] == k+1) break;//break out of second loop
                    if(numpossible>1) break;
                }
                if(numpossible == 1)//if only one possible position
                {
                    //cout<<endl<<"from block search: "<<endl;
                    fill(k+1,row,col);//insert value in correct location in block
                }
            }
        }
    }


}

}

void Puzzle::print()
{
    int i,j,k;

    for(i = 0;i < 9;i++)//i is row
    {
        for(j = 0;j < 9;j++)//j is column
        {
            cout<<board[i][j] << " ";//value then space
            if((j+1) %3 == 0)
            {
                cout << "  ";//extra space between blocks
            }
        }
        cout << endl;
        if((i+1) % 3 == 0)
        {
            if(i == 0){
                j = j/i;
            }
            for(j = 0;j < 9;j++)
            {
                cout<< "  ";//extra space between blocks
            }
            cout << endl;
        }
    }

}

void Puzzle::print3()//prints possible values 3d vector
{
    int i,j,k;
    j = k/0;
```

Value stored to 'j' is never read

```cpp
    for(i = 0;i < 9;i++)
    {
        for(j = 0;j < 9;j++)
        {
            for(k = 0;k < 9;k++)
            {
                cout << possible[i][j][k];
            }
            cout << "   ";
        }
        cout << endl << endl;
    }

}

void Puzzle::updatePossible()
{
    int i,j,k,l,m;

    for(i = 0;i < 9;i++)//i is row
    {
        for(j = 0;j < 9;j++)//j is column
        {
            for(k = 0;k < 9;k++)//k+1 is value
            {
                if(possible[i][j][k]==1)//if it is currently possible
                {
                    for(l = 0;l < 9;l++)//l searches board row and column
                    {
                        if(l == j) continue; //if on the value don't do anything
                        if( board[i][l] == (k+1) )//check row for value
                        {
                            possible[i][j][k] = 0;//set to 0 for not possible
                            break;
                        }
                        if(l == i) continue; //if on the value don't do anything
                        if( board[l][j] == (k+1) )//check column for value
                        {
                            possible[i][j][k] = 0;//set to 0 for not possible
                            break;
                        }
                    }

                    for(l = ( (i/3)*3 );l < ( (i/3)*3+3 );l++)//l is row
                    {
                        for(m = ( (j/3)*3 );m < ( (j/3)*3+3 );m++)// m is coumn
                        {
                            if(l == i && m == j) continue; //if on the actual value don't do anything
                            if( board[l][m] == (k+1) )
                            {
                                possible[i][j][k] = 0;//if a value is found in that 3x3 set not possible (0)
                                break;
                            }
                        }
                    }

                }
            }
        }
    }

}
```

```cpp
void Puzzle::fill(int number, int row, int column)
{
    int z;
    board[row][column] = number;
    //print3();

    for(z = 0;z < 9;z++)//k is value
    {
        possible[row][column][z] = 2;//set all possible to 2 to mean filled
    }
    updatePossible();
    //cout<<number<<" inserted at: "<<row<<","<<column <<endl;
}

int Puzzle::done()
{
    int i,j;
    for(i = 0;i < board.size();i++)
    {
        for(j = 0;j < board[i].size()  ;j++)
        {
            if(board[i][j] == 0) return 0;//if there is an empty value return not done(0)
        }
    }
    return 1;//if it cycles through whole puzzle finding no empty return done(1)
}
```