Cody Constine, Christopher John, Dan Frost & Nolawee Mengist
September 29th 2016
Lab 3
CSCI 3302

Lab 3.1: Mapping

1) The drawback of a data structure that stores the distances between every possible pairs of nodes is that it collects too much information thus wasting memory and other valuable resources. The implementation from the previous question addresses this problem by implementing a heuristic function. We do this by giving priority to nodes that have a lower estimated distance to the goal than others. To do this we would not only mark every node with the actual distance that it took to get there but we'd also account for the estimated cost. An example would be calculating the Euclidean distance between the vertex we are looking at and the goal vertex.

This function assigns an index to the 2d coordinates, and stores them in a struct :

```
void indexAssign(){
  int i,ii;
  int counter = 0;
   for(i=0;i<5;i++)
    {
      for(ii=0;ii<5;ii++)
      {
        index[counter].x = i;
        index[counter].y = ii;
        counter++;
      }
    }
}
```

This functions returns the coordinates from the struct:

```
int getCords(int i)
{
  return index[i].x, index[i].y;
}
```

This function returns a weight based on the position of the coordinates.

```
int getWeight(int s, int e)
{
  int xS, yS, xE, yE;
  xS, yS = getCords(s);
  xE, yE = getCords(e);
  if(grid[xS][yS] == 0 || grid[xE][yE] == 0)
  {
    return 99;
  }
  else if(abs(xS-xE) == 1 && abs(yS-yE) == 1)
  {
    return 1;
  }
  else
  {
    int xDiff = xS-xE;
    int yDiff = yS-yE;
    int i,ii;
    int score = abs(xDiff)+abs(yDiff);
    for(i=1;i<=xDiff;i++)
    {
      for(ii=1;ii<=yDiff;ii++)
      {
        if(grid[i][ii]==0)
        {
          score += 3;
        }
      }
    }
    return score;
  }
}
```