



|  |  |
| --- | --- |
| |  | | --- | | **AB-1** |   **Part (a)**  /\*\* Precondition: the puzzle grid contains the distinct values 0 through side^2 - 1  \* @return true if the tiles in the puzzle are all arranged in increasing order  \* (the hole value 0 may be in any position);  \* false otherwise  \*/  public boolean isDone()  {  int prev = 0;  for (int r = 0; r < side; r++)  {  for (int c = 0; c < side; c++)  {  int num = values[r][c];  if (num != 0)  {  if (num != prev + 1)  return false;  prev = num;  }  }  }  return true;  }  **Part (b)**  /\*\* Initializes the puzzle by placing numbers 0 through side^2 - 1 into random locations  \*/  public void initialize()  {  ArrayList<Integer> temp = new ArrayList<Integer>();  for (int j = 0; j < side \* side; j++)  temp.add(new Integer(j)); 1  for (int r = 0; r < side; r++)  {  for (int c = 0; c < side; c++)  {  int i = (int)(Math.random() \* temp.size());  values[r][c] = temp.get(i).intValue();  temp.remove(i); 2  }  }  }  **Notes:**   1. These three lines were given. 2. Or: 3. values[r][c] = temp.remove(i).getValue();   Or simply  values[r][c] = temp.remove(i);  due to autounboxing.  **Part (c)**  *O*(*n*2) 1  **Notes:**   1. Because the remove method in an ArrayList on average takes *O*(*n*) time.   **Part (d)**  *O*(*n*) |