**Part A:**

public static int[] getCubeTosses(NumberCube cube, int numTosses)

{

int[] cubeTosses = new int[numTosses];

for(int i = 0; i < cubeTosses.length; i++)

{

cubeTosses[i] = cube.toss();

}

return cubeTosses;

}

**Part B:**

public static int getLongestRun(int[] values)

{

int runLength = 0;

int max = 0;

int index = -1;

for(int i = 0; i < values.length - 1; i++)

{

if(values[i] == values[i+1])

{

runLength++;

}

else

{

runLength = 0;

}

if(runLength > max)

{

max = runLength;

index = i - runLength + 1;

}

}

return index;

}

**Alternative solution:**

public static int getLongestRun(int[] values)

{

int maxStart = -1;

int maxLen = -1;

int currentLen = 0;

int currVal = -1;

int currStart = 0;

for(int i = 0; i < values.length; i++)

{

if(values[i] == currVal)

{

currentLen++;

}

else

{

if(currentLen > maxLen)

{

maxLen = currentLen;

maxStart = currStart;

}

currStart = i;

currentLen = 1;

currVal=values[i];

}

}

if(currentLen > maxLen)

{

maxLen = currentLen;

maxStart = currStart;

}

if(maxLen == 1)

return -1;

else

return maxStart;

}

**Second alternative solution:**

public static int getLongestRun(int[] values)

{

int maxLen = 0;

int currLen = 0;

int index = -1;

int currVal = -1;

for (int i = values.length - 1; i >= 0; i--)

{

if (values[i] == currVal)

{

currLen++;

}

else

{

if(maxLen < currLen)

{

maxLen = currLen;

index = i+1;

}

currVal = values[i];

currLen = 1;

}

}

if(maxLen < currLen)

{

maxLen = currLen;

index = 0;

}

if(maxLen == 1)

return -1;

else

return index;

}