Solve

## **SET#A**

| **PYTHON** | **JAVA** |
| --- | --- |
| def findQueenAge(matrix):    row, col = len(matrix), len(matrix[0])    min\_row\_idx = None  min\_row\_sum = 0  min\_col\_idx = None  min\_col\_sum = 0    for idx in range(row):  sum\_values = 0  for val in range(col):  sum\_values += matrix[idx][val]  if min\_row\_sum == 0 or min\_row\_sum > sum\_values:  min\_row\_sum = sum\_values  min\_row\_idx = idx    for idx in range(col):  sum\_values = 0  for val in range(row):  sum\_values += matrix[val][idx]  if min\_col\_sum == 0 or min\_col\_sum > sum\_values:  min\_col\_sum = sum\_values  min\_col\_idx = idx    return matrix[min\_row\_idx][min\_col\_idx] | public static int findQueenAge(int[][] matrix) {  int row = matrix.length;  int col = matrix[0].length;    int minRowIdx = -1;  int minRowSum = Integer.MAX\_VALUE;  int minColIdx = -1;  int minColSum = Integer.MAX\_VALUE;    for (int i = 0; i < row; i++) {  int sumValues = 0;  for (int j = 0; j < col; j++) {  sumValues += matrix[i][j];  }  if (minRowSum > sumValues) {  minRowSum = sumValues;  minRowIdx = i;  }  }    for (int j = 0; j < col; j++) {  int sumValues = 0;  for (int i = 0; i < row; i++) {  sumValues += matrix[i][j];  }  if (minColSum > sumValues) {  minColSum = sumValues;  minColIdx = j;  }  }      return matrix[minRowIdx][minColIdx];  } |

## **RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| Finding array shape (row and column) | 2 |
| Iterate through each row and column for the minimum row index | 2+2 |
| Iterate through each row and column for the minimum column index | 2+2 |
| Finding the minimum row index and minimum column index | 2 |
| Identifying the value at the intersection | 2 |
| Proper Return Statement | 1 |

## **SET#B**

| **PYTHON** | **JAVA** |
| --- | --- |
| def findStrongSource(matrix):    row, col = len(matrix), len(matrix)    max\_row\_idx = None  max\_row\_sum = 0  max\_col\_idx = None  max\_col\_sum = 0    for idx in range(row):  sum\_values = 0  for val in range(col):  sum\_values += matrix[idx][val]  if max\_row\_sum == 0 or max\_row\_sum < sum\_values:  max\_row\_sum = sum\_values  max\_row\_idx = idx    for idx in range(col):  sum\_values = 0  for val in range(row):  sum\_values += matrix[val][idx]  if max\_col\_sum == 0 or max\_col\_sum < sum\_values:  max\_col\_sum = sum\_values  max\_col\_idx = idx    return matrix[max\_row\_idx][max\_col\_idx] | public static int findStrongSource(int[][] matrix) {  int row = matrix.length;  int col = matrix[0].length;  int maxRowIdx = -1;  int maxRowSum = 0;  int maxColIdx = -1;  int maxColSum = 0;  for (int i = 0; i < row; i++) {  int sumValues = 0;  for (int j = 0; j < col; j++) {  sumValues += matrix[i][j];  }  if (maxRowSum == 0 || maxRowSum < sumValues) {  maxRowSum = sumValues;  maxRowIdx = i;  }  }  for (int i = 0; i < col; i++) {  int sumValues = 0;  for (int j = 0; j < row; j++) {  sumValues += matrix[j][i];  }  if (maxColSum == 0 || maxColSum < sumValues) {  maxColSum = sumValues;  maxColIdx = i;  }  }  return matrix[maxRowIdx][maxColIdx];  } |

## **RUBRIC**

| **Category** | **Marks** |
| --- | --- |
| Finding array shape (row and column) | 2 |
| Iterate through each row and column for the maximum row index | 2+2 |
| Iterate through each row and column for the maximum column index | 2+2 |
| Finding the maximum row index and maximum column index | 2 |
| Identifying the value at the intersection | 2 |
| Proper Return Statement | 1 |