Create a program to do the following:

```
[]
1. Create 2d array
2. Find row sum
3. Find col sum
4. Find total sum
5. Quit

Please select 1-5:
```

Depending on the user's choice:

```
if 1. Create 2d array   →   created 2D array 3 rows by 4 columns
                            (fill with random numbers 1-10)
if 2. Find row sum      →   ask row index       → print row sum
if 3. Find col sum      →   ask column index    → print column sum
if 4. Find total sum    →   print total sum
```

Here is a sample run. Your numbers may be different. You should display the 2d_ array by row before displaying the menu as shown below. The first time the array is empty [].

```
[]
1. Create 2d array
2. Find row sum
3. Find col sum
4. Find total sum
5. Quit

Please select 1-5:1

[7, 3, 7, 3]
[6, 1, 9, 7]
[9, 7, 9, 3]
1. Create 2d array
2. Find row sum
3. Find col sum
4. Find total sum
5. Quit

Please select 1-5:2
Enter row index? 0-2: 1
The sum of all row 1 numbers is 23

[7, 3, 7, 3]
[6, 1, 9, 7]
[9, 7, 9, 3]
1. Create 2d array
2. Find row sum
3. Find col sum
4. Find total sum
5. Quit
```

```
Please select 1-5:3
Enter column index? 0-3: 2
The sum of all column 2 numbers is 25

[7, 3, 7, 3]
[6, 1, 9, 7]
[9, 7, 9, 3]
1. Create 2d array
2. Find row sum
3. Find col sum
4. Find total sum
5. Quit

Please select 1-5:4
The total sum is 71

[7, 3, 7, 3]
[6, 1, 9, 7]
[9, 7, 9, 3]
1. Create 2d array
2. Find row sum
3. Find col sum
4. Find total sum
5. Quit

Please select 1-5:5
Good by!
```

Challenge:
Add two more menu options:

```
5. Find row with largest sum
6. Find col with largest sum
7. Quit

Please select 1-7:5
[7, 3, 7, 3]
[6, 1, 9, 7]
[9, 7, 9, 3]
Row 2 has the largest sum 28

Please select 1-7:6
[7, 3, 7, 3]
[6, 1, 9, 7]
[9, 7, 9, 3]
Column 2 has the largest sum 25

Please select 1-7:7
Good by!
```