Wipro coding questions with answers

Telegram :

https://t.me/placements_updates

Ques 1: C Program to check if two given matrices are identical

```
[code language="cpp"]
#include
#define N 4
int areSame(int A[][N], int B[][N])    {
int i, j;
for (i = 0; i < N; i++)
for (j = 0; j < N; j++)
if (A[i][j] != B[i][j])
return 0;
return 1;
}
int main()    {
int A[N][N] = { {1, 1, 1, 1},
{2, 2, 2, 2},
{3, 3, 3, 3},
{4, 4, 4, 4}};
```

int B[N][N] = { {1, 1, 1, 1},

{2, 2, 2, 2},

{3, 3, 3, 3},

{4, 4, 4, 4}};

if (areSame(A, B))

printf(“Matrices are identical”);

Else

printf(“Matrices are not identical”);

return 0;

}

Question 2

: Given a 2D array, print it in spiral form. See the following examples.

```
Input:


        1    2    3    4
        5    6    7    8
        9   10   11   12
        13  14   15   16
Output:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Input:
        1   2    3    4   5    6
        7   8    9   10   11   12
        13  14   15  16   17   18
Output:
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
```

Code:

```
#include <stdio.h>
#define R 3
#define C 6

void spiralPrint (int m, int n, int a[R][C])
```

```
{
  int i, k = 0, l = 0;

/* k - starting row index
m - ending row index
l - starting column index
n - ending column index
i - iterator
*/

  while (k < m && l < n)
      {
/* Print the first row from the remaining rows */
      for (i = l; i < n; ++i)
      {
        printf ("%d ", a[k][i]);
      }
      k++;

/* Print the last column from the remaining columns */
      for (i = k; i < m; ++i)
      {
        printf ("%d ", a[i][n - 1]);
      }
      n--;

/* Print the last row from the remaining rows */
      if (k < m) { for (i = n - 1; i >= l; --i)
          {
            printf ("%d ", a[m - 1][i]);
          }
        m--;
      }

/* Print the first column from the remaining columns */
      if (l < n) { for (i = m - 1; i >= k; --i)
          {
            printf ("%d ", a[i][l]);
          }
        l++;
      }
   }
}

/* Driver program to test above functions */
int main ()
{
  int a[R][C] = { {1, 2, 3, 4, 5, 6},
  {7, 8, 9, 10, 11, 12},
  {13, 14, 15, 16, 17, 18}
  };
  spiralPrint (R, C, a);
  return 0;
}
```

Ques 3:

Given an n-by-n matrix of 0’s and 1’s where all 1’s in each row come before all 0’s, find the most efficient way to return the row with the maximum number of 0’s.

{1,1,1,1},

{1,1,0,0},

{1,0,0,0},

{1,1,0,0},

```
#include <stdio.h>
#include <stdlib.h>
#define COL 4
#define ROW 4
using namespace std;
int main ()
{
  int arr[ROW][COL] = {
    {1, 1, 1, 1},
    {1, 1, 0, 0},
    {1, 0, 0, 0},
    {1, 1, 0, 0},
  };

  int rownum;
  int i = 0, j = COL - 1;
  while (i; 0)
    {
      if (arr[i][j] == 0)
      {
        j--;
        rownum = i;
      }
      else
      i++;
    }
  printf ("%d", rownum);
  getch ();
  return 0;
}
```

Question 4:

A Pythagorean triplet is a set of three integers a, b and c such that $a^2 + b^2 = c^2$. Given a limit, generate all Pythagorean Triples with values smaller than given limit.

```
Input : limit = 20

Output : 3 4 5
         8 6 10
         5 12 13
         15 8 17
         12 16 20
```

A Simple Solution is to generate these triplets smaller than given limit using three nested loop. For every triplet, check if Pythagorean condition is true, if true, then print the triplet. Time complexity of this solution is $O(limit^3)$ where 'limit' is given limit.

```
      a = m2 - n2
       b = 2 * m * n
       c  = m2 + n2
because,
       a2 = m4 + n4 – 2 * m2 * n2
       b2 = 4 * m2 * n2
       c2 = m4 + n4 + 2* m2 * n2
```

We can see that $a^2 + b^2 = c^2$, so instead of iterating for a, b and c we can iterate for m and n and can generate these triplets.

Below is C implementation of above idea.

```
#include <stdio.h>

#include <math.h>

void pythagoreanTriplets(int limit)     {

// triplet:  a^2 + b^2 = c^2

int a, b, c=0;

//  loop from 2 to max_limitit
int m = 2;

// Limiting c would limit all a, b and c

while (c < limit)     {

// now loop on j from 1 to i-1

for (int n = 1; n < m; ++n)     {

// Evaluate and print triplets using

// the relation between a, b and c

a = m*m – n*n;

b = 2*m*n;

c = m*m + n*n;

if (c > limit)

Break;

printf(“%d %d %d\n”, a, b, c);

}

M++;

}

}

// Driver program

int main()   {

int limit = 20;

pythagoreanTriplets(limit);

return 0;

}
```

Output:

```
3 4 5
8 6 10
5 12 13
15 8 17
12 16 20
```