Tutorial 3 – Arrays

1. Explain how the addition of 1 to every element of the two dimensional array ‘array’ is done in the following program. What if the for statement at ‘line a’ is replaced by this statement:

```
add1(array[0], 3 * 4);
```

every element will add 1.
Then, the ~~first four elements will add one again~~

```
#include  <stdio.h>
void  add1(int ar[], int size);
int main()
{
    int  array[3][4];
    int h,k;

    for (h = 0; h < 3; h++)
            for (k = 0; k < 4; k++)
                    scanf(“%d”, &array[h][k]);

    for (h = 0; h < 3; h++)                              /* line a */
            add1(array[h], 4);

    for (h = 0; h < 3; h++) {
            for (k = 0; k < 4; k++)
                    printf(“%10d”, array[h][k]);
            putchar(‘\n’);
    }
    return 0;
}
void add1(int ar[], int size)
{
    int k;

    for (k = 0; k < size; k++)
            ar[k]++;
}
```

The array array[h] is passed using call by reference to function add1 as int ar[] parameter while 4 is passed as int size.

The array is traversed element by element using indexing with ar[k], with k as the index from 0 to n-1. This is done using the for loop. The increment operator then adds 1 to each element after traversing. This changes the element in each column.

Another for loop is used to so the element in each row is also changed.

2. Write a program which will draw the histogram for n integers from 0 to 99. N is input by the user. Each of the n numbers will be generated by calling rand() % 100. The program will consist of two functions (i) to collect the frequency distribution of the numbers (ii) to print the histogram. An example histogram is shown here.

```
0 – 9       |*********************
10 – 19     |************
20 – 29     |*************
30 – 39     |**
.........
90 – 99     |***************
```

```
#include<stdio.h>
#include<stdlib.h>

int N, i, j, c;
int arr[10]={0};
int main(){
   printf("Enter a value of N btw 0 to 10:");
   scanf("%d",&N);
   printf("%d",N);
   printf("\n");

   for(i=0;i<N;i++){
      c = rand()%100;
      arr[i]=c;
      //printf("%d",arr[i]);
      //printf("\n");
   }

   printf("0-9      | ");
   for(i=0;i<arr[0];i++){
      printf("*");}
   printf("\n");
   printf("10-19   | ");
   for(i=0;i<arr[1];i++){
      printf("*");}
   printf("\n");
   printf("20-29   | ");
   for(i=0;i<arr[2];i++){
      printf("*");}
   printf("\n");
   printf("30-39   | ");
   for(i=0;i<arr[3];i++){
      printf("*");}
   printf("\n");
   printf("40-49   | ");
   for(i=0;i<arr[4];i++){
      printf("*");}
   printf("\n");
   printf("50-59   | ");
   for(i=0;i<arr[5];i++){
      printf("*");}
   printf("\n");
   printf("60-69   | ");
   for(i=0;i<arr[6];i++){
      printf("*");}
   printf("\n");
   printf("70-79   | ");
   for(i=0;i<arr[7];i++){
      printf("*");}
   printf("\n");
   printf("80-89   | ");
   for(i=0;i<arr[8];i++){
      printf("*");}
   printf("\n");
   printf("90-99   | ");
   for(i=0;i<arr[9];i++){
      printf("*");}
   printf("\n");

}
```

rowsize = colsize;

void transpose2D(int ar[][SIZE],int rowsize, int colsize);

3. Write a function that takes a square matrix ar, and the array sizes for the rows and columns as parameters, and returns the transpose of the array via call by reference. For example, if the *rowSize* is 4, *colSize* is 4, and the array ar is {1,2,3,4, 5,1,2,2, 6,3,4,4, 7,5,6,7}, then the resultant array will be {1,5,6,7, 2,1,3,5, 3,2,4,6, 4,2,4,7}. That is, for the 4-by-4 matrix:

```
1  2  3  4
5  1  2  2
6  3  4  4
7  5  6  7
```

the resultant array after performing the transpose2D function is:

```
1  5  6  7
2  1  3  5
3  2  4  6
4  2  4  7
```

The function prototype is given below:

void transpose2D(int ar[][SIZE], int rowSize, int colSize);

SIZE is a constant defined at the beginning of the program. For example, #define SIZE 10. The parameters *rowSize* and *colSize* are used to specify the dimensions of the 2-dimensional array (e.g. 4x4) that the function should process.

Write a program to test the function.

```
#include <stdio.h>
int main(){
    int a[3][3] = {1,2,3,4,5,6,7,8,9};
    int b[3][3];
    int j, i;
    for (i=0;i<3;i++){
        for (j=0;j<3;j++){
            b[j][i]=a[i][j];
        }
    }
    for (i=0;i<3;i++){
        for (j=0;j<3;j++){
            printf("%d",b[i][j]);}
        printf("\n");
    }
    return 0;
}
```

replace 3 with rowsize\

4. A square matrix (2-dimensional array of equal dimensions) can be reduced to upper-triangular form by setting each diagonal element to the sum of the original elements in that column and setting to 0s all the elements below the diagonal. For example, the 4-by-4 matrix:

```
4  3  8  6
9  0  6  5
5  1  2  4
9  8  3  7
```

would be reduced to

```
27  3  8  6
 0  9  6  5
 0  0  5  4
 0  0  0  7
```

void reducematrix2d(int ar[][size], int rowsize, int colsize);

replace 3 with rowsize, rowsize=colsize

Write a function reduceMatrix2D() to reduce a matrix with dimensions of *rowSize* and *colSize*. The prototype of the function is:

void reduceMatrix2D(int ar[][SIZE], int rowSize, int colSize);

SIZE is a constant defined at the beginning of the program. For example, #define SIZE 10. The parameters rowSize and colSize are used to specify the dimensions of the 2-dimensional array (e.g. 4x4) that the function should process.

Write a program to test the function.

```
#include <stdio.h>
int main(){
    int a[3][3] = {1,2,3,40,5,6,77,88,9};
    int b[3][3];
    int j, i,n,t;
    n = a[0][0];
    for (i=0;i<3;i++){
        for(j=i;j<3;j++){
            if(i+j<3){
            a[i][i]+=a[i+j][i];
            t = a[0][0];
        }}}

    for (i=0;i<3;i++){
        for(j=i;j<3;j++){
            if(i+j<3){
            a[i+j][i]=0;
        }}}
    a[0][0] =t-n;

    for (i=0;i<3;i++){
        for (j=0;j<3;j++){
            printf("%d ",a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```