Embedded C Programming – Laboratory (L33+L34)

Experiment 3 – Programs on Array

Task 1.01:

Array and calling the function by value

```
#include <stdio.h>
void display ( int ) ;//first function declaration - 1st stage
int main( )/*second function declaration
this main() remains asleep until the display() function completes it job*/
{
     int i ;
     int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
          for (i = 0 ; i <= 6 ; i++)
          display (marks[i]) ;//function call - 3rd stage
return 0 ;
}
void display (int m)//function definition - 2nd stage
{
printf ("%d",m) ;
```

Output:

```
55 65 75 56 78 78 90
--------------------------------
```

Task 1.02:

Array and calling the function by reference

```
#include <stdio.h>
void display (int *);
int main( )
```

```
{
     int i ;
     int marks[ ] = { 55, 65, 75, 56, 78, 78, 90 } ;
          for (i = 0 ; i <= 6 ; i++)
          display ( &marks[ i ] ) ;
return 0 ;
}
void display ( int *m )
{
printf ( "%d ", *m ) ;
}
```

Output:

```
55 65 75 56 78 78 90
--------------------------------
```

Task 1.03:

Write a C code to get the average marks of 10 students. Marks should be greater than 0 and less than 100. Use an one dimensional array to store the student's marks. E.g.: marks[i].

Solution:

```
# include <stdio.h>
int main( )
{
int avg, sum = 0 ;
int i ;
int marks[10] ; /* array declaration */
for ( i = 0 ; i <= 9 ; i++ )
{
```

```
     do{
     printf ( "Enter marks " ) ;
     scanf ( "%d", &marks[ i ] ) ; /* store data in array
*/
          if(marks[i]<0||marks[i]>100){
          printf ( "Enter valid marks\n " ) ;
          }
     }while(marks[i]<0||marks[i]>100);
}
for ( i = 0 ; i <= 9 ; i++ )
     sum = sum + marks[ i ] ; /* read data from an array*/
     avg = sum / 10 ;
     printf("Sum = %d",sum);
     printf ("Average marks = %d\n", avg ) ;
return 0 ;

}
```

Output:

```
Enter marks 70
Enter marks 90
Enter marks 200
Enter valid marks
 Enter marks 100
Enter marks 50
Enter marks 60
Enter marks -23
Enter valid marks
 Enter marks 99
Enter marks 89
Enter marks 98
Enter marks 97
Enter marks 99
Sum = 852Average marks = 85
```

Task 1.04:

Get the roll number along with the marks in 2D array as shown in the output.

```
#include <stdio.h>

int main() {

    char stud[4][10]; // row-4:columnAssuming roll no. is a string of up to 9 characters

    int marks[4][2];//row-4:column-2

    int i;

    for (i = 0; i <= 3; i++) {

        printf("Enter roll no. and marks: ");

        scanf("%s %d %d", stud[i], &marks[i][0],&marks[i][1]);

    }

    for (i = 0; i <= 3; i++) {

        printf("%s %d %d\n", stud[i], marks[i][0], marks[i][1]);

    }

    return 0;

}
```

Output:

```
Enter roll no. and marks: 20bec1111 77 88
Enter roll no. and marks: 20bec1112 88 99
Enter roll no. and marks: 20bec1113 99 100
Enter roll no. and marks: 20bec1114 98 100
20bec1111 77 88
20bec1112 88 99
20bec1113 99 100
20bec1114 98 100
```

Task 1.05:

//Use a 3D-array to get patient's index, number of visits, and metric count. Maximum patients should be 3, visits should be 2 and metric count are SBP, DBP, and HR. If the threshold is greater than 100, give an emergency message as shown in the ouput.

```
#include <stdio.h>

#define PATIENT_COUNT 3

#define VISIT_COUNT 2

#define METRIC_COUNT 3

void checkEmergency(float
patients[][VISIT_COUNT][METRIC_COUNT], int patientIndex) {

    float threshold = 100.0; // Example threshold for an
emergency

    int visit,metric;

    for (visit = 0; visit < VISIT_COUNT; visit++) {

        for (metric = 0; metric < METRIC_COUNT; metric++)
{

            if (patients[patientIndex][visit][metric] >
threshold) {

                printf("Emergency Message: Patient %d,
Visit %d has a critical value in metric %d!\n",
patientIndex + 1, visit + 1, metric + 1);

                return;

            }

        }

    }

}

int main() {
```

```
    float
patients[PATIENT_COUNT][VISIT_COUNT][METRIC_COUNT] = {
        {
            {110.0, 70.0, 80.0}, // Patient 1, Visit 1
metrics: SBP, DBP, HR
            {120.0, 80.0, 90.0}  // Patient 1, Visit 2
metrics
        },
        {
            {90.0, 60.0, 100.0}, // Patient 2, Visit 1
metrics
            {95.0, 65.0, 105.0}  // Patient 2, Visit 2
metrics
        },
        {
            {120.0, 80.0, 120.0}, // Patient 3, Visit 1
metrics
            {125.0, 85.0, 130.0}  // Patient 3, Visit 2
metrics
        }
    };
    // Simulate checking for emergencies for each patient
and visit
    int i;
    for (i = 0; i < PATIENT_COUNT; i++) {
        checkEmergency(patients, i);
    }
    return 0;
}
```

```
Emergency Message: Patient 1, Visit 1 has a critical value in metric 1!
Emergency Message: Patient 2, Visit 2 has a critical value in metric 3!
Emergency Message: Patient 3, Visit 1 has a critical value in metric 1!
```