

National Institute of Technology Karnataka Surathkal,  
Mangalore - 575025



---

DEPARTMENT OF INFORMATION TECHNOLOGY  
DEPARTMENT OF CIVIL ENGINEERING

---

## LAB ASSIGNMENT 1

Submitted for Parallel Computing (IT301) By

Gaurav Chaurasia

181CV155

To

Dr. Geetha V

*Dept of IT, NITK Surathkal*

[Code link](#)  
[github](#)

# Problem - 1

Generating random values and setting up the temp array

```

69
70 ~ int main() {
71 ~     // vector<vector<vector<int>>> tem(1000, vector<vector<int>>>
72 ~     // setting up the array with random data between 1 to 100;
73 ~     for (int i = 0; i < CITY; i++) {
74 ~         for (int j = 0; j < HOUR; j++) {
75 ~             for (int k = 0; k < READINGS; k++) {
76 ~                 tem[i][j][k] = 1 + (rand() % 100);
77 ~             }
78 ~         }
79 ~     }
80
81 ~     for (int i = 0; i < CITY; i++) {
82 ~         for (int j = 0; j < HOUR; j++) {
83 ~             for (int k = 0; k < READINGS; k++) {
84 ~                 mean_tem[i][j] += tem[i][j][k];
85 ~             }
86 ~             mean_tem[i][j] /= READINGS;
87 ~         }
88 ~     }
89
90 ~     min_temperature(7);
91 ~     max_temperature(7);
92 ~     avg_temperature(7);
93 ~     return 0;
94 ~ }

```

## Function to find max temp

```

46
47 ~ void max_temperature(int count=CITY) {
48 ~     cout << "#####\tCITY NO-----MAX TEMPERATURE\t#####\n" << endl;
49 ~     for (int i = 0; i < min(count, CITY); i++) {
50 ~         int max_tem = INT_MIN;
51 ~         for (int j = 0; j < HOUR; j++) {
52 ~             max_tem = max(max_tem, mean_tem[i][j]);
53 ~         }
54 ~         cout << "[ 'CITY NO': " << i+1 << ", 'MAX TEMPERATURE': " << max_tem << " ]" << endl;
55 ~     }
56 ~ }
57

```

## Function to find avg temp

```

57
58 v void avg_temperature(int count=CITY) {
59     cout << "#####\tCITY NO----AVG TEMPERATURE\t#####" << endl;
60 v     for (int i = 0; i < min(count, CITY); i++) {
61         float avg_tem = 0;
62 v         for (int j = 0; j < HOUR; j++) {
63             avg_tem += mean_tem[i][j];
64         }
65         avg_tem /= HOUR;
66         cout << "[ 'CITY NO': " << i+1 << ", 'AVG TEMPERATURE': " << avg_tem << " ]" << endl;
67     }
68 }
69

```

## Function to find max temp

```

35
36 v void min_temperature(int count=CITY) {
37     cout << "#####\tCITY NO----MIN TEMPERATURE\t#####" << endl;
38 v     for (int i = 0; i < min(count, CITY); i++) {
39         int min_tem = INT_MAX;
40 v         for (int j = 0; j < HOUR; j++) {
41             min_tem = min(min_tem, mean_tem[i][j]);
42         }
43         cout << "[ 'CITY NO': " << i+1 << ", 'MIN TEMPERATURE': " << min_tem << " ]" << endl;
44     }
45 }
46

```

## Function to print temp info

```

22 // prints the temperature information
23 v void prind_temperature_info() {
24 v     for (int i = 0; i < CITY; i++) {
25 v         for (int j = 0; j < HOUR; j++) {
26             cout << "[ ";
27 v             for (int k = 0; k < READINGS; k++) {
28                 cout << tem[i][j][k] << " ";
29             }
30             cout << "]" << endl;
31         }
32         cout << endl;
33     }
34 }

```

## Function to generate random nums

```
11
12 // generate count number of random nums between 1 to 100
13 void genrate_random_nums(int count) {
14     int result;
15     for (int i = 0; i < count; i++) {
16         result = 1 + (rand() % 100);
17         cout << result << " ";
18     }
19     cout << endl;
20 }
21
```

## Printing temp info of first 7 cities

```
> g++ temperature.cpp
> ./a.out
***** CITY NO-----MIN TEMPERATURE *****
['CITY NO': 1, 'MIN TEMPERATURE': 31 ]
['CITY NO': 2, 'MIN TEMPERATURE': 31 ]
['CITY NO': 3, 'MIN TEMPERATURE': 37 ]
['CITY NO': 4, 'MIN TEMPERATURE': 28 ]
['CITY NO': 5, 'MIN TEMPERATURE': 38 ]
['CITY NO': 6, 'MIN TEMPERATURE': 27 ]
['CITY NO': 7, 'MIN TEMPERATURE': 34 ]
***** CITY NO-----MAX TEMPERATURE *****
['CITY NO': 1, 'MAX TEMPERATURE': 65 ]
['CITY NO': 2, 'MAX TEMPERATURE': 74 ]
['CITY NO': 3, 'MAX TEMPERATURE': 67 ]
['CITY NO': 4, 'MAX TEMPERATURE': 74 ]
['CITY NO': 5, 'MAX TEMPERATURE': 74 ]
['CITY NO': 6, 'MAX TEMPERATURE': 66 ]
['CITY NO': 7, 'MAX TEMPERATURE': 70 ]
***** CITY NO-----AVG TEMPERATURE *****
['CITY NO': 1, 'AVG TEMPERATURE': 50.0417 ]
['CITY NO': 2, 'AVG TEMPERATURE': 49.2083 ]
['CITY NO': 3, 'AVG TEMPERATURE': 51.1667 ]
['CITY NO': 4, 'AVG TEMPERATURE': 51.5 ]
['CITY NO': 5, 'AVG TEMPERATURE': 53.5 ]
['CITY NO': 6, 'AVG TEMPERATURE': 50.6667 ]
['CITY NO': 7, 'AVG TEMPERATURE': 50.375 ]

C:\mnt\d\SEM\SEM 06\IT301-Parallel Computing\IT301-LAB\LAB-1 11:03:18 AM
```

## Code for Problem 1

```
#include <iostream>
#include <climits>
#include <vector>
using namespace std;

#define CITY 1000
#define HOUR 24
#define READINGS 10
int tem[CITY][HOUR][READINGS];
int mean_tem[CITY][HOUR];

// generate count number of random numers between 1 to 100
void genrate_random_nums(int count) {
    int result;
    for (int i = 0; i < count; i++) {
        result = 1 + (rand() % 100);
        cout << result << " ";
    }
    cout << endl;
}

// prints the temperature information
void prind_temperature_info() {
    for (int i = 0; i < CITY; i++) {
        for (int j = 0; j < HOUR; j++) {
            cout << "[";
            for (int k = 0; k < READINGS; k++) {
                cout << tem[i][j][k] << " ";
            }
            cout << "]" << endl;
        }
        cout << endl;
    }
}
```

```

void min_temperature(int count=CITY) {
    cout << "#####\tCITY NO-----MIN TEMPERATURE\t#####" <<
endl;
    for (int i = 0; i < min(count, CITY); i++) {
        int min_tem = INT_MAX;
        for (int j = 0; j < HOUR; j++) {
            min_tem = min(min_tem, mean_tem[i][j]);
        }
        cout << "[ 'CITY NO': " << i+1 << ", 'MIN
TEMPERATURE': " << min_tem << " ]" << endl;
    }
}

```

```

void max_temperature(int count=CITY) {
    cout << "#####\tCITY NO-----MAX TEMPERATURE\t#####" <<
endl;
    for (int i = 0; i < min(count, CITY); i++) {
        int max_tem = INT_MIN;
        for (int j = 0; j < HOUR; j++) {
            max_tem = max(max_tem, mean_tem[i][j]);
        }
        cout << "[ 'CITY NO': " << i+1 << ", 'MAX
TEMPERATURE': " << max_tem << " ]" << endl;
    }
}

```

```

void avg_temperature(int count=CITY) {
    cout << "#####\tCITY NO-----AVG TEMPERATURE\t#####" <<
endl;
    for (int i = 0; i < min(count, CITY); i++) {
        float avg_tem = 0;
        for (int j = 0; j < HOUR; j++) {
            avg_tem += mean_tem[i][j];
        }
        avg_tem /= HOUR;
        cout << "[ 'CITY NO': " << i+1 << ", 'AVG
TEMPERATURE': " << avg_tem << " ]" << endl;
    }
}

```

```
    }  
}  
  
int main() {  
    // vector<vector<vector<int>>> tem(1000,  
vector<vector<int>>(24, vector<int>(10, 0)));  
    // setting up the array with random data between 1 to  
100;  
    for (int i = 0; i < CITY; i++) {  
        for (int j = 0; j < HOUR; j++) {  
            for (int k = 0; k < READINGS; k++) {  
                tem[i][j][k] = 1 + (rand() % 100);  
            }  
        }  
    }  
  
    for (int i = 0; i < CITY; i++) {  
        for (int j = 0; j < HOUR; j++) {  
            for (int k = 0; k < READINGS; k++) {  
                mean_tem[i][j] += tem[i][j][k];  
            }  
            mean_tem[i][j] /= READINGS;  
        }  
    }  
  
    min_temperature(7);  
    max_temperature(7);  
    avg_temperature(7);  
    return 0;  
}
```

## Problem 2

```

39     }
40 }
41 ~int main() {
42     generate_hex();
43
44     // setting up the array with random data between 0 to 255;
45 ~    for (int i = 0; i < GRID; i++) {
46 ~        for (int j = 0; j < GRID; j++) {
47             pixle[i][j] = (rand() % 256);
48         }
49     }
50     prind_pixle_grid(20, 20);
51     inc_pixle(20);
52     return 0;
53 }

```

```

33
34 void inc_pixle(int count) {
35     for (int i = 0; i < GRID; i++) {
36         for (int j = 0; j < GRID; j++) {
37             pixle[i][j] = min(255, pixle[i][j] + count);
38         }
39     }
40 }

```

```

23
24 // prints the temperature information
25 void prind_pixle_grid(int max_row=GRID, int max_col=GRID) {
26     for (int i = 0; i < min(max_row, GRID); i++) {
27         for (int j = 0; j < min(max_col, GRID); j++) {
28             // ... (code for printing temperature information) ...
29         }
30     }
31 }
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```



# Printing one part of 256X256 GRID

## First 25X25 GRID

```

> g++ pixle.cpp
> ./a.out
07 06 69 73 51 FF 4A EC 29 CD BA AB F2 FB E3 46 7C C2 54 F8
08 70 D4 B2 8A 29 54 48 9A 0A BC D5 0E 18 A8 44 AC 5B F3 8E
95 AA 32 CA 6C 49 AE 90 CD 16 88 BA AC 7A A6 F2 B4 A8 CA 99
F3 58 46 06 47 28 26 0E 00 D2 EB B2 1F 8C 3A 38 C6 54 2A AB
28 8C B6 87 1B 64 F5 61 AB 1C E7 90 58 90 1E E5 02 A8 11 77
90 EF FD A5 2B 40 64 55 42 35 AB 33 71 38 E2 CF DC 8D 62 2B
A1 01 E4 D9 A8 59 25 31 C7 9A 4C 7A 89 A7 2D AA 6A F2 44 F8
78 D6 2C DB FD 22 8C C7 D3 88 90 B0 80 56 48 AC 68 3F 2F 3E
B7 F4 88 79 2C F0 6D 84 FE 91 42 4D 64 60 44 86 C9 A2 D9 66
CA 0C E4 CE 02 8D A9 40 62 F5 93 FF 42 08 2E C9 D6 76 05 D8
22 B3 30 3E C2 58 FB 12 7C B7 98 CA 14 A9 7D 63 A7 B7 2B 95
B7 4C 31 6F 88 7D 18 F8 AA B7 B4 41 39 B2 B5 85 03 C2 CC F6
C1 57 E0 30 8C 05 DD EF 9D 7D 17 A5 BC A6 28 9D 34 3E B3 EA
30 0A 16 1B 0C 70 7F 7E D9 11 F0 57 E9 89 68 DD 35 FB DA 1A
A5 08 DF 56 95 34 3E 96 D2 66 22 D8 00 EE F1 54 B0 AB 36 A8
9E A7 09 88 ED 7F 30 85 51 93 44 68 68 0F D6 5A 2D E4 16 B7
B1 D4 58 59 DA D8 B3 5A 18 5E 9F 40 80 64 C7 51 E4 B7 6E 27
F7 D7 A0 A5 D2 C6 53 4B 6A 30 DD 9A B0 19 B7 0F 36 CF F5 B9
E4 31 FB 60 C1 AC BD D5 E7 BE 24 ED 23 87 95 23 86 85 D4 7F
CF FE 59 29 E9 0A 8F D7 EE 02 4B 38 80 D1 9D 4A 85 AC 90 95

```

## Code for Problem 2

```

#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

#define GRID 256
int pixle[GRID][GRID];
vector<string> pixle_hex; // for storing hex conversion
// 00, 01, 02, .... , FD, FE, FF

void generate_hex() {
    char hex[16] = { '0', '1', '2', '3', '4', '5', '6',
        '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };
    for (int i = 0; i < 16; i++) {
        for (int j = 0; j < 16; j++) {
            string str = "\0";
            str.push_back(hex[i]);
            str.push_back(hex[j]);
            pixle_hex.push_back(str);
        }
    }
}

```

```

        }
    }
    sort(pixle_hex.begin(), pixle_hex.end());
}

// prints the temperature information
void prind_pixle_grid(int max_row=GRID, int max_col=GRID) {
    for (int i = 0; i < min(max_row, GRID); i++) {
        for (int j = 0; j < min(max_col, GRID); j++) {
            cout << pixle_hex[pixle[i][j]] << " ";
        }
        cout << endl;
    }
}

void inc_pixle(int count) {
    for (int i = 0; i < GRID; i++) {
        for (int j = 0; j < GRID; j++) {
            pixle[i][j] = min(255, pixle[i][j] + count);
        }
    }
}

int main() {
    generate_hex();

    // setting up the array with random data between 0 to
    255;
    for (int i = 0; i < GRID; i++) {
        for (int j = 0; j < GRID; j++) {
            pixle[i][j] = (rand() % 256);
        }
    }
    prind_pixle_grid(20, 20);
    inc_pixle(20);
    return 0;
}

```