```c
/*Design a program to log temperature readings from multiple sensors for
24 hours, sampled every hour.
Requirements:
Use a 2D array of size [N][24] to store temperature data, where N is the
number of sensors (defined as a const variable).
Use static variables to calculate and store the daily average temperature
for each sensor.
Use nested for loops to populate and analyze the array.
Use if statements to identify sensors exceeding a critical threshold
temperature.
*/
#include<stdio.h>
#define MAX 3
#define T 70
void main()
{
    const int max=MAX,t=T;
    int temp[MAX][24];
    static int avg[MAX];
    int sum=0;
    srand(time(NULL));
    printf("Enter the temperature readings\n");
    for(int i=0;i<max;i++)
    {
        printf("The temperature data for sensor %d \n",i+1);
        for(int j=0;j<24;j++)
        {
            temp[i][j] = (rand()%100)+1;
            printf("%d ",temp[i][j]);
        }
        printf("\n");
    }
    for(int i=0;i<max;i++)
    {
        sum=0;
        for(int j=0;j<24;j++)
        {
            sum+=temp[i][j];
        }
        avg[i]=sum/24;
```

```
    }
    for(int i=0;i<max;i++)
    for(int j=0;j<24;j++)
    {
        if(temp[i][j]>t)
        printf("The temperature at sensor %d exceeds threshhold limits :
%d\n",i+1,temp[i][j]);
    }



}
```

```
Enter the temperature readings
The temperature data for sensor 1
20 10 43 26 84 37 42 37 28 74 35 98 37 80 75 85 19 33 92 89 21 86 51 84
The temperature data for sensor 2
36 91 69 46 69 83 56 64 18 44 7 42 55 50 16 79 73 25 93 79 42 8 85 82
The temperature data for sensor 3
94 59 13 7 74 61 75 3 91 66 21 11 5 11 92 43 33 40 86 36 23 48 79 84
The temperature at sensor 1 exceeds threshhold limits : 84
The temperature at sensor 1 exceeds threshhold limits : 74
The temperature at sensor 1 exceeds threshhold limits : 98
The temperature at sensor 1 exceeds threshhold limits : 80
The temperature at sensor 1 exceeds threshhold limits : 75
The temperature at sensor 1 exceeds threshhold limits : 85
The temperature at sensor 1 exceeds threshhold limits : 92
The temperature at sensor 1 exceeds threshhold limits : 89
The temperature at sensor 1 exceeds threshhold limits : 86
The temperature at sensor 1 exceeds threshhold limits : 84
The temperature at sensor 2 exceeds threshhold limits : 91
The temperature at sensor 2 exceeds threshhold limits : 83
The temperature at sensor 2 exceeds threshhold limits : 79
The temperature at sensor 2 exceeds threshhold limits : 73
The temperature at sensor 2 exceeds threshhold limits : 93
The temperature at sensor 2 exceeds threshhold limits : 79
The temperature at sensor 2 exceeds threshhold limits : 85
The temperature at sensor 2 exceeds threshhold limits : 82
The temperature at sensor 3 exceeds threshhold limits : 94
The temperature at sensor 3 exceeds threshhold limits : 74
The temperature at sensor 3 exceeds threshhold limits : 75
The temperature at sensor 3 exceeds threshhold limits : 91
The temperature at sensor 3 exceeds threshhold limits : 92
The temperature at sensor 3 exceeds threshhold limits : 86
The temperature at sensor 3 exceeds threshhold limits : 79
The temperature at sensor 3 exceeds threshhold limits : 84
```

```
/*Simulate the control of an LED matrix of size 8x8. Each cell in the
matrix can be ON (1) or OFF (0).
Requirements:
Use a 2D array to represent the LED matrix.
Use static variables to count the number of ON LEDs.
Use nested for loops to toggle the state of specific LEDs based on input
commands.
Use if statements to validate commands (e.g., row and column indices).
```

```
*/
#include<stdio.h>
#define MAX 8
void main()
{
const int max=MAX;
int LED[MAX][MAX],row,col,input;
static int on=0;
srand(time(NULL));
for(int i=0;i<max;i++)
{
    for(int  j=0;j<max;j++)
    {
        LED[i][j]=rand()%2;
        printf("LED[%d][%d] %d ",i,j,LED[i][j]);
        if(LED[i][j]==1)
        on++;
    }
    printf("\n");
}
printf("\n");
printf("Number of ON LEDs is %d\n",on);
printf("Enter the row,column and input\n");
scanf("%d %d %d",&row,&col,&input);
for(int i=0;i<max;i++)
for(int j=0;j<max;j++)
{
    if(i==row && j==col)
    {
        if(LED[row][col]!=input)
        {
            if(input==1)
            {
                LED[row][col]=input;
                on++;
            }
            else
            {
                LED[row][col]=input;
                on--;
```

```
            }
        }
    }
}
for(int i=0;i<max;i++)
{   for(int j=0;j<max;j++)
    printf("LED[%d][%d] %d ",i,j,LED[i][j]);
    printf("\n");
}
printf("Number of ON LEDs is %d\n",on);
}
```

```
                ^~~~
LED[0][0] 0 LED[0][1] 0 LED[0][2] 1 LED[0][3] 0 LED[0][4] 0 LED[0][5] 1 LED[0][6] 0 LED[0][7] 1
LED[1][0] 0 LED[1][1] 0 LED[1][2] 0 LED[1][3] 1 LED[1][4] 1 LED[1][5] 1 LED[1][6] 0 LED[1][7] 1
LED[2][0] 1 LED[2][1] 1 LED[2][2] 1 LED[2][3] 0 LED[2][4] 0 LED[2][5] 0 LED[2][6] 1 LED[2][7] 1
LED[3][0] 1 LED[3][1] 1 LED[3][2] 1 LED[3][3] 0 LED[3][4] 1 LED[3][5] 0 LED[3][6] 0 LED[3][7] 1
LED[4][0] 0 LED[4][1] 0 LED[4][2] 1 LED[4][3] 0 LED[4][4] 0 LED[4][5] 0 LED[4][6] 1 LED[4][7] 1
LED[5][0] 1 LED[5][1] 0 LED[5][2] 0 LED[5][3] 0 LED[5][4] 0 LED[5][5] 1 LED[5][6] 1 LED[5][7] 1
LED[6][0] 0 LED[6][1] 0 LED[6][2] 0 LED[6][3] 0 LED[6][4] 1 LED[6][5] 0 LED[6][6] 1 LED[6][7] 0
LED[7][0] 1 LED[7][1] 1 LED[7][2] 0 LED[7][3] 1 LED[7][4] 1 LED[7][5] 0 LED[7][6] 1 LED[7][7] 1

Number of ON LEDs is 32
Enter the row,column and input
0 3
1
LED[0][0] 0 LED[0][1] 0 LED[0][2] 1 LED[0][3] 1 LED[0][4] 0 LED[0][5] 1 LED[0][6] 0 LED[0][7] 1
LED[1][0] 0 LED[1][1] 0 LED[1][2] 0 LED[1][3] 1 LED[1][4] 1 LED[1][5] 1 LED[1][6] 0 LED[1][7] 1
LED[2][0] 1 LED[2][1] 1 LED[2][2] 1 LED[2][3] 0 LED[2][4] 0 LED[2][5] 0 LED[2][6] 1 LED[2][7] 1
LED[3][0] 1 LED[3][1] 1 LED[3][2] 1 LED[3][3] 0 LED[3][4] 1 LED[3][5] 0 LED[3][6] 0 LED[3][7] 1
LED[4][0] 0 LED[4][1] 0 LED[4][2] 1 LED[4][3] 0 LED[4][4] 0 LED[4][5] 0 LED[4][6] 1 LED[4][7] 1
LED[5][0] 1 LED[5][1] 0 LED[5][2] 0 LED[5][3] 0 LED[5][4] 0 LED[5][5] 1 LED[5][6] 1 LED[5][7] 1
LED[6][0] 0 LED[6][1] 0 LED[6][2] 0 LED[6][3] 0 LED[6][4] 1 LED[6][5] 0 LED[6][6] 1 LED[6][7] 0
LED[7][0] 1 LED[7][1] 1 LED[7][2] 0 LED[7][3] 1 LED[7][4] 1 LED[7][5] 0 LED[7][6] 1 LED[7][7] 1
Number of ON LEDs is 33
PS D:\projects\quest\C>
```

```
/* Track the movement of a robot on a grid of size M x N.
Requirements:
Use a 2D array to store visited positions (1 for visited, 0 otherwise).
Declare grid dimensions using const variables.
Use a while loop to update the robot's position based on input directions
(e.g., UP, DOWN, LEFT, RIGHT).
Use if statements to ensure the robot stays within bounds.
 */
#include<stdio.h>
#include<stdlib.h>
#define M 4
```

```c
#define N 3
void main()
{
    const int m=M;
    const int n =N;
    int grid[M][N]={0},c;
    int i,j;
    printf("Enter the starting point of the robot\n");
    scanf("%d %d",&i,&j);
    grid[i][j]=1;
    for(int x=0;x<m;x++)
    {
        for(int y=0;y<n;y++)
        printf("%d ",grid[x][y]);
        printf("\n");

    }
   while (1)
   {
        printf("Enter direction\n");
        printf("1.UP\n");
        printf("2.DOWN\n");
        printf("3.LEFT\n");
        printf("4.RIGHT\n");
        printf("5.EXIT\n");
        scanf("%d",&c);
        switch (c)
        {
            case 1: if((i-1)<0)
                        printf("The location will be out of bounds\n");
                    else
                        {   if(grid[i-1][j]==0)
                            {   i-=1;
                                grid[i][j]=1;
                            }
                            else
                                printf("Alreday visited block\n");
                        }
                     for(int x=0;x<m;x++)
                        {
```

```
                    for(int y=0;y<n;y++)
                    printf("%d ",grid[x][y]);
                    printf("\n");
                }
        break;
        case 2: if((i+1)>=m)
                printf("The location will be out of bounds\n");
            else
                {
                    if(grid[i+1][j]==0)
                    {
                        i+=1;
                        grid[i][j]=1;
                    }
                    else
                        printf("Already visted block\n");
                }
             for(int x=0;x<m;x++)
                {
                    for(int y=0;y<n;y++)
                    printf("%d ",grid[x][y]);
                    printf("\n");
                }
        break;
        case 3: if((j-1)<0)
                printf("The location will be out of bounds\n");
            else
                {   if(grid[i][j-1]==0)
                    {   j-=1;
                        grid[i][j]=1;
                    }
                    else
                        printf("Alreday visited block\n");
                }
             for(int x=0;x<m;x++)
                {
                    for(int y=0;y<n;y++)
                    printf("%d ",grid[x][y]);
                    printf("\n");
                }
```

```
            break;
            case 4: if((j+1)>=n)
                        printf("The location will be out of bounds\n");
                    else
                        {
                            if(grid[i][j+1]==0)
                            {
                                j+=1;
                                grid[i][j]=1;
                            }
                            else
                                printf("Already visted block\n");
                        }
                    for(int x=0;x<m;x++)
                        {
                            for(int y=0;y<n;y++)
                            printf("%d ",grid[x][y]);
                            printf("\n");
                        }
            break;
            case 5: exit(0);
            break;
        }

    }


}
```

```
Enter the starting point of the robot
1 1
0 0 0
0 1 0
0 0 0
0 0 0
Enter direction
1.UP
2.DOWN
3.LEFT
4.RIGHT
5.EXIT
1
0 1 0
0 1 0
0 0 0
0 0 0
Enter direction
1.UP
2.DOWN
3.LEFT
4.RIGHT
5.EXIT
3
1 1 0
0 1 0
0 0 0
0 0 0
Enter direction
1.UP
2.DOWN
3.LEFT
4.RIGHT
5.EXIT
2
1 1 0
1 1 0
```

```c
/*Store and analyze data from multiple sensors placed in a 3D grid (e.g., environmental sensors in a greenhouse).
Requirements:
Use a 3D array of size [X][Y][Z] to store data, where dimensions are defined using const variables.
Use nested for loops to populate the array with sensor readings.
Use if statements to find and count sensors reporting critical values (e.g., temperature > 50°C).
Use static variables to store aggregated results (e.g., average readings per layer).
*/
#include<stdio.h>
#define X 2
#define Y 3
#define Z 4
void main()
{
    const int x=X,y=Y,z=Z;
    int temp[X][Y][Z],sum,avg[X];
    int count=0;
    srand(time(NULL));
    printf("Collecting data from environment\n");
    for(int i=0;i<x;i++)
    for(int j=0;j<y;j++)
    for(int k=0;k<z;k++)
    temp[i][j][k]=(rand()/100)+1;
    for(int i=0;i<x;i++)
    {
        sum=0;
        for(int j=0;j<y;j++)
        for(int k=0;k<z;k++)
        {
            sum+=temp[i][j][k];
            if(temp[i][j][k]>50)
            count++;
        }
        avg[i]= sum/(y*z);
    }
    printf("The average temperature for each layer\n");
    for(int i=0;i<x;i++)
```

```
    printf("Avg for layer %d : %d\n",i+1,avg[i]);
    printf("Number of sensors exceeding the thresshold %d\n",count);

}
```

```
Collecting data from environment
The average temperature for each layer
Avg for layer 1 : 177
Avg for layer 2 : 147
Number of sensors exceeding the thresshold 20
PS D:\projects\quest\C>
```

```
/*Perform edge detection on a grayscale image represented as a 2D array.
Requirements:
Use a 2D array of size [H][W] to store pixel intensity values (defined
using const variables).
Use nested for loops to apply a basic filter (e.g., Sobel filter) on the
matrix.
Use decision-making statements to identify and highlight edge pixels
(threshold-based).
Store the output image in a static 2D array.
*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define H 5
#define W 5
#define THRESHOLD 100
int main() {
    const int height = H;
    const int width = W;
    int sumX,sumY,magnitude;
    static int edge[H][W];
    int image[H][W] = {
        {200, 200, 200, 200, 200},
        {200, 0, 0, 0, 200},
        {200, 0, 255, 0, 200},
        {200, 0, 0, 0, 200},
        {200, 200, 200, 200, 200}
    };
     int Gx[3][3] = {
```

```
        {-1, 0, 1},
        {-2, 0, 2},
        {-1, 0, 1}
    };

    int Gy[3][3] = {
        {-1, -2, -1},
        {0, 0, 0},
        {1, 2, 1}
    };
    for (int i = 1; i < height - 1; i++) {
        for (int j = 1; j < width - 1; j++) {
            sumX = 0;
            sumY = 0;
            for (int k = -1; k <= 1; k++) {
                for (int l = -1; l <= 1; l++) {
                    sumX += image[i + k][j + l] * Gx[k + 1][l + 1];
                    sumY += image[i + k][j + l] * Gy[k + 1][l + 1];
                }
            }
            magnitude = (int)sqrt((sumX * sumX) + (sumY * sumY));
            if (magnitude > THRESHOLD) {
                edge[i][j] = 255;
            } else {
                edge[i][j] = 0;
            }
        }
    }
    printf("Original Image:\n");
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            printf("%3d ", image[i][j]);
        }
        printf("\n");
    }
    printf("\nEdge Detected Image:\n");
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            printf("%3d ", edge[i][j]);
        }
```

```
        printf("\n");
    }

    return 0;
}
```

```
Original Image:
200 200 200 200 200
200   0   0   0 200
200   0 255   0 200
200   0   0   0 200
200 200 200 200 200

Edge Detected Image:
  0   0   0   0   0
  0 255 255 255   0
  0 255   0 255   0
  0 255 255 255   0
  0   0   0   0   0
```

```
/*Manage the states of traffic lights at an intersection with four roads,
each having three lights (red, yellow, green).
Requirements:
Use a 2D array of size [4][3] to store the state of each light (1 for ON,
0 for OFF).
Use nested for loops to toggle light states based on time intervals.
Use static variables to keep track of the current state cycle.
Use if statements to validate light transitions (e.g., green should not
overlap with red).
*/
#include <stdio.h>
#include <unistd.h>
#define R 4
#define L 3
int RED=0,YELLOW=1,GREEN=2;
void print_traffic_lights(int traffic_lights[R][L]) {
    const char *light_names[] = { "RED", "YELLOW", "GREEN" };
    printf("\nTraffic Light States:\n");
    for (int i = 0; i < R; i++) {
```

```
        printf("Road %d: ", i + 1);
        for (int j = 0; j < L; j++) {
            if (traffic_lights[i][j] == 1) {
                printf("%s ", light_names[j]);
            } else {
                printf("OFF ");
            }
        }
        printf("\n");
    }
}
void update_traffic_lights(int traffic_lights[R][L], int cycle) {
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < L; j++) {
            traffic_lights[i][j] = 0;
        }
        traffic_lights[i][RED] = 1;
    for (int i = 0; i < R; i++) {
        switch (cycle) {
            case 0:if (i == 0)
            {
                    traffic_lights[i][RED] = 0;
                    traffic_lights[i][GREEN] = 1;
            }
                break;
            case 1: if(i == 0)
            {
                traffic_lights[i][GREEN] = 0;
                traffic_lights[i][YELLOW] = 1;
            }
                break;
            case 3: if (i == 1) {
                    traffic_lights[i][RED] = 0;
                    traffic_lights[i][GREEN] = 1;
                }
                break;

            case 2:  if (i == 1) {
                    traffic_lights[i][GREEN] = 0;
                    traffic_lights[i][YELLOW] = 1;
```

```
                }
                break;

            case 4: if(i == 2)
            {
                    traffic_lights[i][RED] = 0;
                    traffic_lights[i][GREEN] = 1;
            }
            break;

            case 5: if (i == 2) {
                    traffic_lights[i][GREEN] = 0;
                    traffic_lights[i][YELLOW] = 1;
                }
                break;
            case 6:if(i == 3)
            {
                    traffic_lights[i][RED] = 0;
                    traffic_lights[i][GREEN] = 1;
            }
            break;
            case 7:
            if(i == 3)
            {
                    traffic_lights[i][RED] = 0;
                    traffic_lights[i][GREEN] = 1;
            }
            break;

        }
    }
}
}
void main() {
    const int r = R;
    const int l = L;
    static int traffic_lights[R][L] = { {1, 0, 0}, {1, 0, 0}, {1, 0, 0},
{1, 0, 0} };
    static int cycle = 0;
    while (1) {
```

```
        print_traffic_lights(traffic_lights);
        sleep(1);
        cycle++;
        if (cycle >= 8) {
            cycle = 0;
        }
        update_traffic_lights(traffic_lights, cycle);
    }
}
```

```
Traffic Light States:
Road 1: RED OFF OFF
Road 2: RED OFF OFF
Road 3: RED OFF OFF
Road 4: RED OFF OFF

Traffic Light States:
Road 1: RED YELLOW OFF
Road 2: RED OFF OFF
Road 3: RED OFF OFF
Road 4: RED OFF OFF

Traffic Light States:
Road 1: RED OFF OFF
Road 2: RED YELLOW OFF
Road 3: RED OFF OFF
Road 4: RED OFF OFF

Traffic Light States:
Road 1: RED OFF OFF
Road 2: OFF OFF GREEN
Road 3: RED OFF OFF
Road 4: RED OFF OFF

Traffic Light States:
Road 1: RED OFF OFF
Road 2: RED OFF OFF
Road 3: OFF OFF GREEN
Road 4: RED OFF OFF

Traffic Light States:
Road 1: RED OFF OFF
Road 2: RED OFF OFF
Road 3: RED YELLOW OFF
Road 4: RED OFF OFF
```

```
/*Simulate an animation on an LED cube of size 4x4x4.
Requirements:
Use a 3D array to represent the LED cube's state.
```

```c
/*Use nested for loops to turn ON/OFF LEDs in a predefined pattern.
Use static variables to store animation progress and frame counters.
Use if-else statements to create transitions between animation frames.
*/
#include<stdio.h>
#define S 4
const int s=S;

void print(int cube[S][S][S])
{
    for(int i=0;i<s;i++)
    {
        for(int j=0;j<s;j++)
        {
            for(int k=0;k<s;k++)
            printf("%d ",cube[i][j][k]);
            printf("\n");
        }
        printf("\n");
    }
}
void update(int cube[S][S][S],int frame)
{
    static int progress=0;
    for(int i=0;i<s;i++)
    for(int j=0;j<s;j++)
    for(int k=0;k<s;k++)
    cube[i][j][k]=0;
    if(progress==0)
    {
        for(int i=0;i<s;i++)
        cube[i][i][i]=1;
    }
    else if(progress==1)
    {
        for(int i=0;i<s;i++)
        for(int j=0;j<S;j++)
        for(int k=0;k<s;k++)
        cube[i][j][k]=1;
    }
```

```
    else if(progress==2)
    {
        for(int i=0;i<s;i++)
        for(int j=0;j<s;j++)
        cube[i][j][j]=1;
    }
    else if(progress ==3)
    {
        for(int i=0;i<s;i++)
        for(int j=0;j<s;j++)
        cube[0][i][j]=1;
    }
    if(frame%2==0)
        progress =(progress+1)%4;

}
void main()
{
    static int frame=0;
    int cube[S][S][S]={{{0}}};
    while(1)
    {
        printf("Display cube\n");
        print(cube);
        sleep(1);
        frame++;
        update(cube,frame);
    }

}
```

```
Display cube
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

Display cube
1 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

0 0 0 0
0 1 0 0
0 0 0 0
0 0 0 0

0 0 0 0
0 0 0 0
0 0 1 0
```

```
/*Track inventory levels for multiple products stored in a 3D warehouse
(e.g., rows, columns, and levels).
```

```c
/*
Requirements:
Use a 3D array of size [P][R][C] to represent the inventory of P products
in a grid.
Use nested for loops to update inventory levels based on shipments.
Use if statements to detect low-stock levels in any location.
Use a static variable to store total inventory counts for each product.
*/
#include <stdio.h>
#define PRODUCTS 3
#define ROWS 4
#define COLUMNS 4
const int products = PRODUCTS;
const int rows = ROWS;
const int columns = COLUMNS;
void print_inventory(int warehouse[PRODUCTS][ROWS][COLUMNS])
{
    printf("\nUpdated Inventory Levels:\n");
    for (int p = 0; p < products; p++) {
        printf("Product %d:\n", p + 1);
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < columns; c++) {
                printf("%d ", warehouse[p][r][c]);
            }
            printf("\n");
        }
        printf("\n");
    }
}
void update_inventory(int warehouse[PRODUCTS][ROWS][COLUMNS], int
shipments[PRODUCTS][ROWS][COLUMNS])
{
    for (int p = 0; p < products; p++) {
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < columns; c++) {
                warehouse[p][r][c] += shipments[p][r][c];
            }
        }
    }
}
void check_low_stock(int warehouse[PRODUCTS][ROWS][COLUMNS],int threshold)
```

```
{
    for (int p=0;p<products;p++)
    {
        static int total_inventory=0;
        total_inventory=0;
        for (int r = 0; r < rows; r++) {
            for (int c = 0; c < columns; c++) {
                total_inventory+=warehouse[p][r][c];
                if (warehouse[p][r][c]<threshold) {
                    printf("Low stock alert for Product %d at [%d][%d]: %d
units\n",p + 1,r + 1,c + 1,warehouse[p][r][c]);
                }
            }
        }
        printf("Total inventory for Product %d: %d units\n",p +
1,total_inventory);
    }
}
void main() {
    int threshold=10;
    static int warehouse[PRODUCTS][ROWS][COLUMNS]={{{0}}};
    int shipments[PRODUCTS][ROWS][COLUMNS] = {
        {
            {10, 20, 30, 40},
            {15, 25, 35, 45},
            {20, 30, 40, 50},
            {25, 35, 45, 55}
        },
        {
            {5, 10, 15, 20},
            {10, 15, 20, 25},
            {15, 20, 25, 30},
            {20, 25, 30, 35}
        },
        {
            {2, 4, 6, 8},
            {3, 6, 9, 12},
            {4, 8, 12, 16},
            {5, 10, 15, 20}
        }
```

```
    };
    update_inventory(warehouse,shipments);
    print_inventory(warehouse);
    check_low_stock(warehouse,threshold);
}
```

```
Updated Inventory Levels:
Product 1:
10 20 30 40
15 25 35 45
20 30 40 50
25 35 45 55

Product 2:
5 10 15 20
10 15 20 25
15 20 25 30
20 25 30 35

Product 3:
2 4 6 8
3 6 9 12
4 8 12 16
5 10 15 20

Total inventory for Product 1: 520 units
Low stock alert for Product 2 at [1][1]: 5 units
Total inventory for Product 2: 320 units
Low stock alert for Product 3 at [1][1]: 2 units
Low stock alert for Product 3 at [1][2]: 4 units
Low stock alert for Product 3 at [1][3]: 6 units
Low stock alert for Product 3 at [1][4]: 8 units
Low stock alert for Product 3 at [2][1]: 3 units
Low stock alert for Product 3 at [2][2]: 6 units
Low stock alert for Product 3 at [2][3]: 9 units
Low stock alert for Product 3 at [3][1]: 4 units
Low stock alert for Product 3 at [3][2]: 8 units
Low stock alert for Product 3 at [4][1]: 5 units
Total inventory for Product 3: 140 units
```

```
/*Apply a basic signal filter to a 3D matrix representing sampled signals
over time.
Requirements:
Use a 3D array of size [X][Y][Z] to store signal data.
Use nested for loops to apply a filter that smoothens the signal values.
```

```c
Use if statements to handle boundary conditions while processing the
matrix.
Store the filtered results in a static 3D array.
*/
#include<stdio.h>
#define X 3
#define Y 3
#define Z 3
void print(int signal[X][Y][Z])
{
    const int x=X,y=Y,z=Z;
    for(int i=0;i<x;i++)
    {
        for(int j=0;j<y;j++)
        {
            for(int k=0;k<z;k++)
            printf("%d ",signal[i][j][k]);
            printf("\n");
        }
        printf("\n");
    }
}
void filter(int signal[X][Y][Z],int out[X][Y][Z])
{
    const int x=X,y=Y,z=Z;
    int sum,count,ni,nj,nk;
    for(int i=0;i<x;i++)
    for(int j=0;j<y;j++)
    for(int k=0;k<z;k++)
        {
            sum=0;
            count=0;
            for(int di=-1;di<=1;di++)
            for(int dj=-1;dj<=1;dj++)
            for(int dk=-1;dk<=1;dk++)
            {
                ni=i+di;
                nj=j+dj;
                nk=k+dk;
                if(ni>=0 && ni<x && nj>=0 && nj<y && nk>=0 && nk<z)
```

```
                {
                    sum+=signal[ni][nj][nk];
                    count++;
                }
            }
            out[i][j][k]=sum/count;
        }
}
void main()
{
    const int x=X,y=Y,z=Z;
    int signal[X][Y][Z],out[X][Y][Z]={{{0}}};
    int t=1;
    for(int i=0;i<x;i++)
    for(int j=0;j<y;j++)
    for(int k=0;k<z;k++)
    {
        signal[i][j][k]=t;
        t++;
    }
    printf("The input signal is\n");
    print(signal);
    filter(signal,out);
    printf("The output signal is \n");
    print(out);
}
```

```
The input signal is
1 2 3
4 5 6
7 8 9

10 11 12
13 14 15
16 17 18

19 20 21
22 23 24
25 26 27

The output signal is
7 8 8
9 9 10
10 11 11

12 12 13
13 14 14
15 15 16

16 17 17
18 18 19
19 20 20
```

```c
/*Analyze weather data recorded over multiple locations and days, with
hourly samples for each day.
Requirements:
Use a 3D array of size [D][L][H] to store temperature readings (D days, L
locations, H hours per day).
Use nested for loops to calculate the average daily temperature for each
location.
Use if statements to find the location and day with the highest
temperature.
Use static variables to store results for each location.
*/
#include<stdio.h>
#define D 3
```

```
#define L 3
#define H 3
void main()
{
    const int d=D,l=L,h=H;
    int temp[D][L][H]={{{0}}};
    int avt[D][L]={{0}};
    static int location[L]={0};
    int sum,high=0,day,loc,hou,count;
    srand(time(NULL));
    for(int i=0;i<D;i++)
    for(int j=0;j<L;j++)
    for(int k=0;k<H;k++)
    temp[i][j][k]=(rand()%100)+1;
    for(int i=0;i<D;i++)
    for(int j=0;j<L;j++)
    {
        sum=0;
        for(int k=0;k<H;k++)
        sum+=temp[i][j][k];
        avt[i][j]=sum/h;
    }
    printf("The average daily temeprature\n");
    for(int i=0;i<D;i++)
    {
        sum=0;
        count=0;
        for(int j=0;j<L;j++)
        for(int k=0;k<H;k++)
        {
            sum+=temp[i][j][k];
            count++;
        }
        printf("Daily avg temperature at location %d is
%d\n",i+1,sum/count);

    }

    for(int i=0;i<D;i++)
    for(int j=0;j<L;j++)
```

```
    for(int k=0;k<H;k++)
    {
        if(temp[i][j][k]>high)
        {
            day=i;
            loc=j;
            high=temp[i][j][k];
        }

    }
    printf("The location %d on day %d has the highest temperature of
%d\n",loc,day,high);


}
```

```
The average daily temeprature
Daily avg temperature at location 1 is 74
Daily avg temperature at location 2 is 63
Daily avg temperature at location 3 is 48
The location 2 on day 1 has the highest temperature of 97
```