```c
/*
OpenMP program to multiply two matrices.
*/

#include <stdio.h>
#include <omp.h>
int main()
{
    int A[3][3] = {1, 1, 1,
                   1, 1, 1,
                   1, 1, 1};
    int B[3][3] = {2, 2, 2,
                   2, 2, 2,
                   2, 2, 2};
    int C[3][3], i, j, k;
#pragma omp parallel for shared(C)
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            C[i][j] = 0;
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
#pragma omp parallel for shared(C, B, A)
            for (k = 0; k < 3; k++)
                C[i][j] = C[i][j] + A[i][k] * B[k][j];
    for (i = 0; i < 3; i++)
    {
        printf("\n");
        for (j = 0; j < 3; j++)
            printf(" %d", C[i][j]);
    }
    fgetc(stdin);
}

/*OpenMP program to find the dot product, cross
product of two vectors.*/

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

void display_vector(int *result, int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d\t", result[i]);
    printf("\n");
}
int dot_product(int *first, int *second, int size)
{
    int result = 0, i;
#pragma omp parallel for reduction(+ \
                                   : result)
    for (i = 0; i < size; i++)
        result += first[i] * second[i];
    return result;
}
void cross_product(int *first, int *second, int size)
{
    int *result = (int *)malloc(size * sizeof(int));
    int i, j, m;
```

```
61
62      omp_set_dynamic(0);
63      m = omp_get_num_procs();
64      omp_set_num_threads(m);
65
66  #pragma omp parallel for shared(result, first, second)
67      for (i = 0; i < size; i++)
68          result[i] = first[(i + 1) % size] * second[(i + 2) % size] - first[(i +
    2) % size] * second[(i + 1) % size];
69      display_vector(result, size);
70  }
71  int main()
72  {
73      int size = 3;
74      int first[3] = {3, -5, 4};
75      int second[3] = {2, 6, 5};
76      printf("\n Result after dot product : %d", dot_product(first, second,
    size));
77      printf("\n Result after vector product : ");
78      cross_product(first, second, size);
79      return 0;
80  }
81
82  /*
83  OpenMP program to find the determinant of a
84  3x3 matrix.
85  */
86
87  #include <stdio.h>
88  #include <math.h>
89  #include <stdlib.h>
90  #include <omp.h>
91
92  int **allocate_mem(int **temp)
93  {
94      temp = (int **)malloc(2 * sizeof(int *));
95      for (size_t i = 0; i < 2; i++)
96      {
97          temp[i] = (int *)malloc(2 * sizeof(int));
98      }
99      return temp;
100 }
101 void deallocate_mem(int **temp)
102 {
103     for (size_t i = 0; i < 2; i++)
104     {
105         free(temp[i]);
106     }
107     free(temp);
108 }
109 void display_matrix(int input_matrix[3][3])
110 {
111     for (size_t i = 0; i < 3; i++)
112     {
113         for (size_t j = 0; i < 3; i++)
114         {
115             printf("%d\t", input_matrix[i][j]);
116         }
117         printf("\n");
118     }
```

```c
}
int **get_cofactor(int input_matrix[3][3], int selected)
{
    int row = 0, col = 0;
    int **temp = NULL;
    temp = allocate_mem(temp);
    for (size_t i = 1; i < 3; i++)
    {
        for (size_t j = 0; j < 3; i++)
        {
            if (j ≠ selected)
            {
                temp[row][col++] = input_matrix[i][j];
            }
        }
        row++;
        col = 0;
    }
    return temp;
}
int get_minor(int input_matrix[3][3], int selected)
{
    int **temp = NULL, result = 0;
    temp = allocate_mem(temp);
    temp = get_cofactor(input_matrix, selected);
    result = temp[0][0] * temp[1][1] - temp[1][0] * temp[0][1];
    deallocate_mem(temp);
    return result;
}

int get_determinant(int input_matrix[3][3])
{
    int i, result = 0;
#pragma omp parallel for shared(input_matrix) reduction(+ \
                                                        : result)
    for (i = 0; i < 3; i++)
        result += (int)pow(-1, i) * input_matrix[0][i] *
get_minor(input_matrix, i);
    return result;
}

int main(int argc, char *argv[])
{
    int input_matrix[3][3];
    int counter = 0;
    for (size_t i = 0; i < 3; i++)
    {
        for (size_t j = 0; j < 3; i++)
        {
            input_matrix[i][j] = atoi(argv[counter++]);
        }
    }
    printf("Determinant of the matrix : %d", get_determinant(input_matrix));
    return 0;
}

```