**Task No. 1:** Write a multithreaded OpenMP C program for performing summation of numbers.

**Solution:**

#include <stdio.h>

#include <stdlib.h>

#include <omp.h>

int main() {

int n;

printf("Enter the number of elements: ");

scanf("%d", &n);

// Create an array of numbers.

int \*numbers = malloc(sizeof(int) \* n);

for (int i = 0; i < n; i++) {

numbers[i] = i;

}

// Calculate the sum of the numbers in parallel.

int sum = 0;

#pragma omp parallel for reduction(+:sum)

for (int i = 0; i < n; i++) {

printf("Thread %d adds %d to sum\n", omp\_get\_thread\_num(), numbers[i]);

sum += numbers[i];

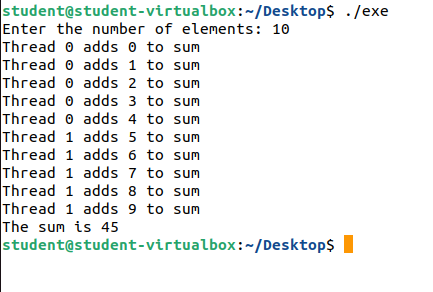
}

// Print the sum.

printf("The sum is %d\n", sum);

free(numbers);

return 0;

}

**Output:**

**Task No. 2:** Write a OpenMP C program in which you have a range of numbers, and you want to find all the even numbers in the range. The search operation can be parallelized using OpenMP to improve performance.

**Solution:**

#include <stdio.h>

#include <omp.h>

int main() {

int start = 1;

int end = 50;

int count = 0;

#pragma omp parallel num\_threads(4) reduction(+:count)

{

int thread\_num = omp\_get\_thread\_num();

int thread\_count = 0;

#pragma omp for

for (int i = start; i <= end; i++) {

if (i % 2 == 0) {

thread\_count++;

printf("Thread %d added: %d\n", thread\_num, i);

}

}

count += thread\_count;

}

printf("Total even numbers: %d\n", count);

return 0;

}

**Output:**

