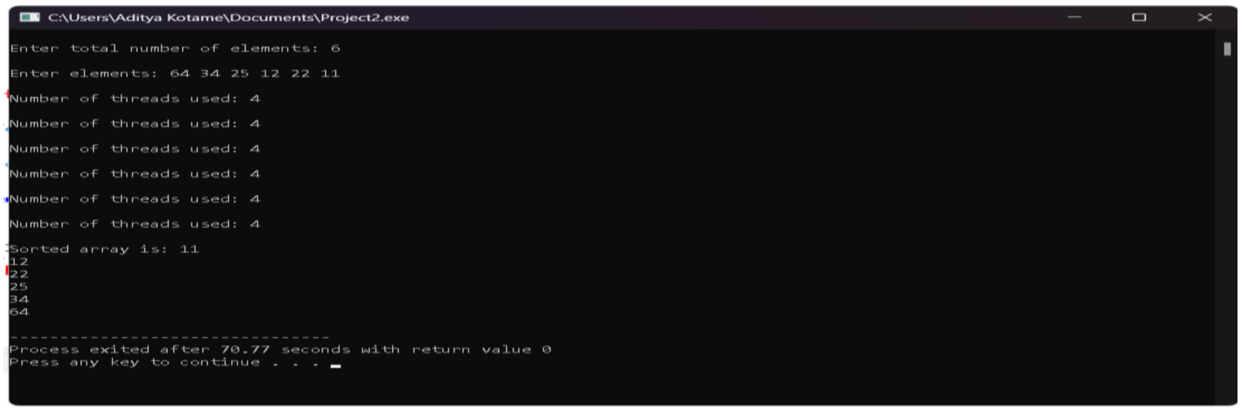
Parallel_Bubble_Sort.cpp

```
#include<iostream>
#include<stdlib.h>
#include<omp.h>
using namespace std;

void bubble(int *, int);
void swap(int &, int &);
void bubble(int *a, int n)
{
    for (int i = 0; i < n; i++) {
        int first = i % 2;
        // Parallelized for odd/even indexed comparison
        #pragma omp parallel
        {
            // Print the number of threads once inside the parallel region
            #pragma omp single
            {
                cout << "\nNumber of threads used: " << omp_get_num_threads() << endl;
            }
            #pragma omp for
            for (int j = first; j < n - 1; j += 2) {
                if (a[j] > a[j + 1]) {
                    swap(a[j], a[j + 1]);
                }
            }
        }
    }
}
void swap(int &a, int &b)
{
    int temp = a;
    a = b;
    b = temp;
}
int main()
{
    int *a, n;
    cout << "\nEnter total number of elements: ";
    cin >> n;
    a = new int[n];
    cout << "\nEnter elements: ";
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    bubble(a, n);
    cout << "\nSorted array is: ";
    for (int i = 0; i < n; i++) {
        cout << a[i] << endl;
    }
    delete[] a;  // Free dynamically allocated memory
    return 0;
}
```

Output:



C:\Users\Aditya Kotame\Documents\Project2.exe

```
Enter total number of elements: 6
Enter elements: 64 34 25 12 22 11
Number of threads used: 4
Number of threads used: 4
Number of threads used: 4
Number of threads used: 4
Number of threads used: 4
Number of threads used: 4
Sorted array is: 11
12
22
25
34
64
--------------------------------
Process exited after 70.77 seconds with return value 0
Press any key to continue . . .
```