HPC

Assignment no 1:

Code:

BFS

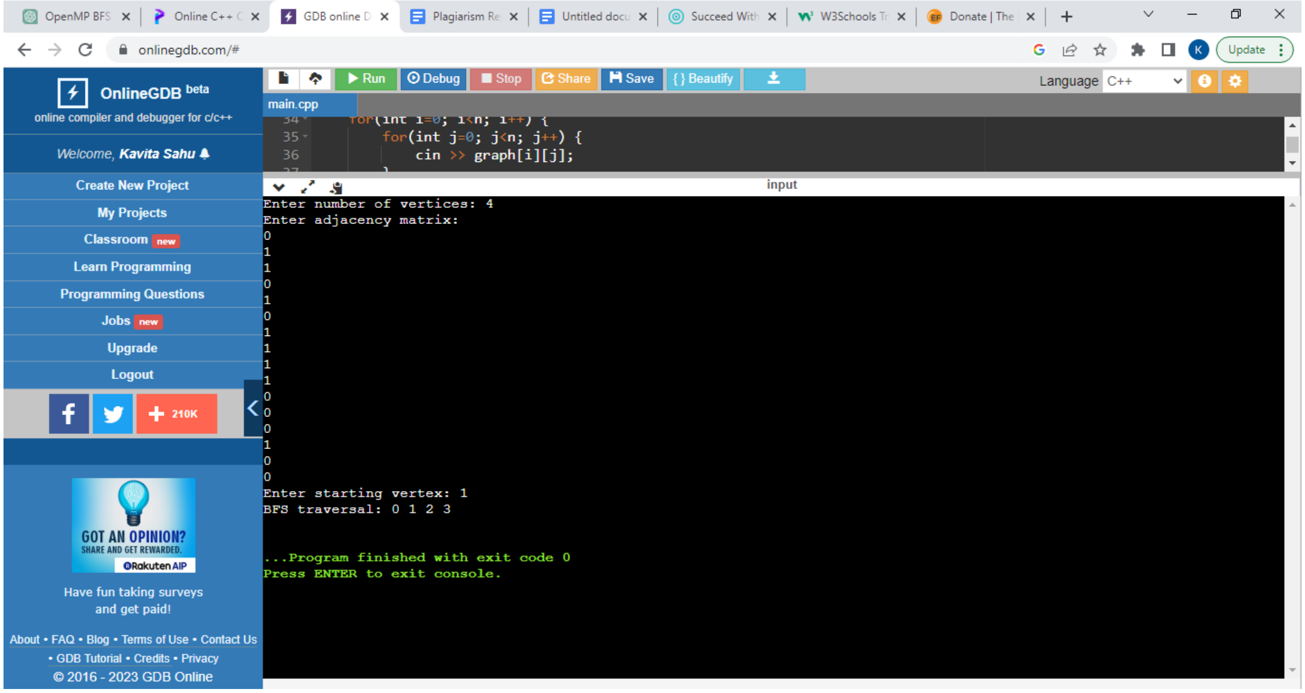
```
// Online C++ compiler to run C++ program online
#include <iostream>
#include <queue>
#include <omp.h>

using namespace std;

const int MAX = 1000;
int graph[MAX][MAX], visited[MAX];

void bfs(int start, int n) {
    queue<int> q;
    visited[start] = 1;
    q.push(start);

    while(!q.empty()) {
        int curr = q.front();
        q.pop();
        #pragma omp parallel for shared(graph, visited, q) schedule(dynamic)
        for(int i=0; i<n; i++) {
            if(graph[curr][i] && !visited[i]) {
                visited[i] = 1;
                q.push(i);
            }
        }
    }
}

int main() {
    int n, start;
    cout << "Enter number of vertices: ";
    cin >> n;
    cout << "Enter adjacency matrix:\n";
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            cin >> graph[i][j];
        }
    }
    cout << "Enter starting vertex: ";
    cin >> start;

    #pragma omp parallel num_threads(4)
    {
        bfs(start, n);
    }

    cout << "BFS traversal: ";
    for(int i=0; i<n; i++) {
        if(visited[i])
            cout << i << " ";
    }
    cout << endl;
    return 0;
}
```

DFS:

```
/******************************************************************************

Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS,
SQLite, Prolog.
Code, Compile, Run and Debug online from anywhere in world.

*******************************************************************************/
#include <iostream>
#include <stack>
#include <omp.h>

using namespace std;

const int MAX = 1000;
int graph[MAX][MAX], visited[MAX];

void dfs(int start, int n) {
    stack<int> s;
    s.push(start);

    while(!s.empty()) {
        int curr = s.top();
        s.pop();
        if(!visited[curr]) {
            visited[curr] = 1;
            #pragma omp parallel for shared(graph, visited, s) schedule(dynamic)
            for(int i=0; i<n; i++) {
                if(graph[curr][i] && !visited[i]) {
                    s.push(i);
                }
            }
        }
    }
}

int main() {
    int n, start;
    cout << "Enter number of vertices: ";
    cin >> n;
    cout << "Enter adjacency matrix:\n";
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            cin >> graph[i][j];
        }
    }
    cout << "Enter starting vertex: ";
    cin >> start;

    #pragma omp parallel num_threads(4)
    {
        dfs(start, n);
    }

    cout << "DFS traversal: ";
    for(int i=0; i<n; i++) {
        if(visited[i])
            cout << i << " ";
    }
    cout << endl;
    return 0;
}
```

Output:

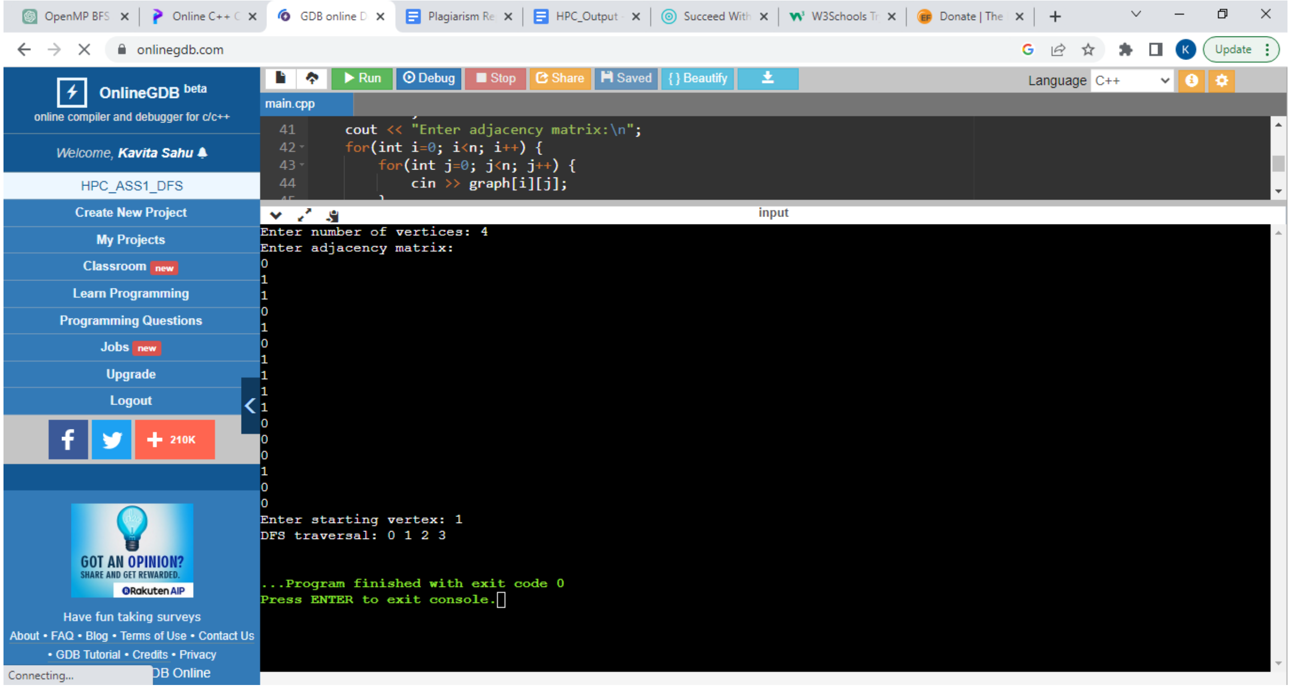
BFS:



```
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            cin >> graph[i][j];
        }
```

```
Enter number of vertices: 4
Enter adjacency matrix:
0
1
1
0
1
0
1
1
1
1
0
0
0
1
0
0
Enter starting vertex: 1
BFS traversal: 0 1 2 3


...Program finished with exit code 0
Press ENTER to exit console.
```

DFS:



```
    cout << "Enter adjacency matrix:\n";
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            cin >> graph[i][j];
```

```
Enter number of vertices: 4
Enter adjacency matrix:
0
1
1
0
1
0
1
1
1
1
0
0
0
1
0
0
Enter starting vertex: 1
DFS traversal: 0 1 2 3


...Program finished with exit code 0
Press ENTER to exit console.
```

Assignment 2

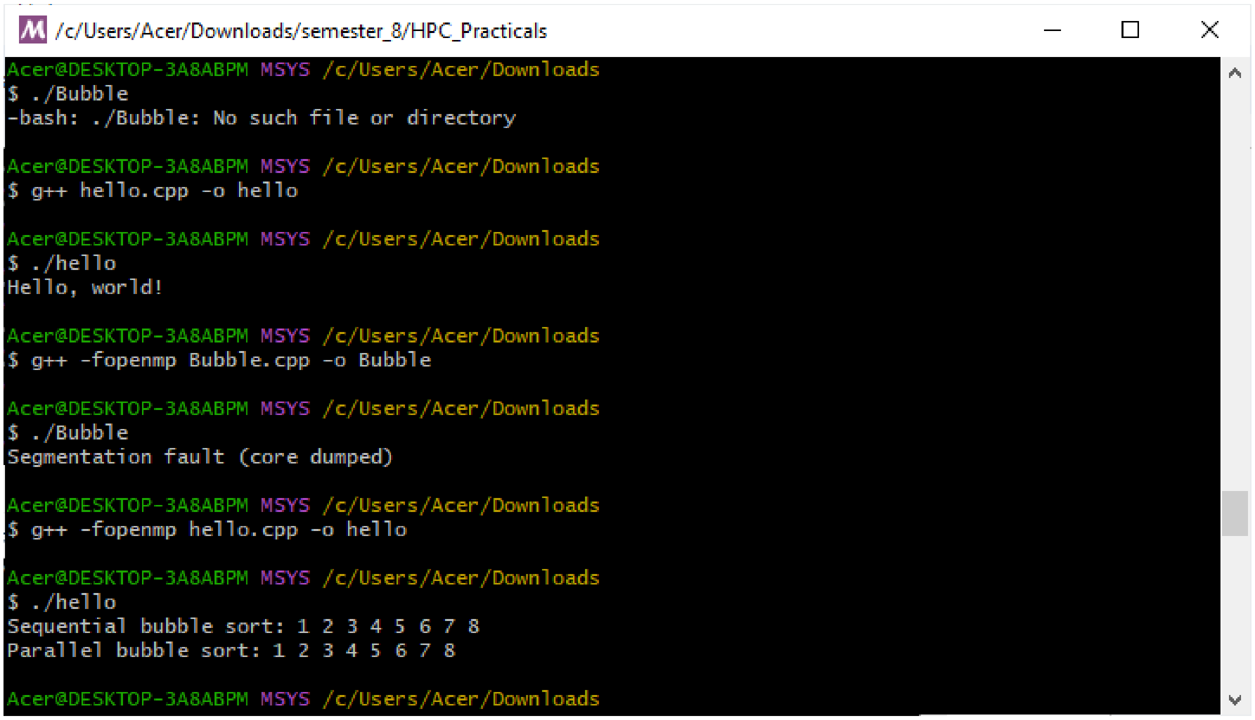
Code:
Bubble sort:

```
#include <iostream>
#include <vector>
#include <omp.h>

using namespace std;

void bubbleSort(vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                swap(arr[j], arr[j + 1]);
            }
        }
    }
}

void parallelBubbleSort(vector<int>& arr) {
    int n = arr.size();
    int num_threads = omp_get_num_threads();
    int chunk_size = n / num_threads;

    #pragma omp parallel shared(arr) num_threads(num_threads)
    {
        #pragma omp for schedule(static, chunk_size)
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    swap(arr[j], arr[j + 1]);
                }
            }
        }
    }
}

int main() {
    vector<int> arr = {5, 3, 8, 6, 7, 2, 1, 4};

    // Sequential bubble sort
    bubbleSort(arr);
    cout << "Sequential bubble sort: ";
    for (int x : arr) {
        cout << x << " ";
    }
    cout << endl;

    // Parallel bubble sort
    arr = {5, 3, 8, 6, 7, 2, 1, 4};
    omp_set_num_threads(4);
    parallelBubbleSort(arr);
    cout << "Parallel bubble sort: ";
    for (int x : arr) {
        cout << x << " ";
    }
    cout << endl;

    return 0;
}
```

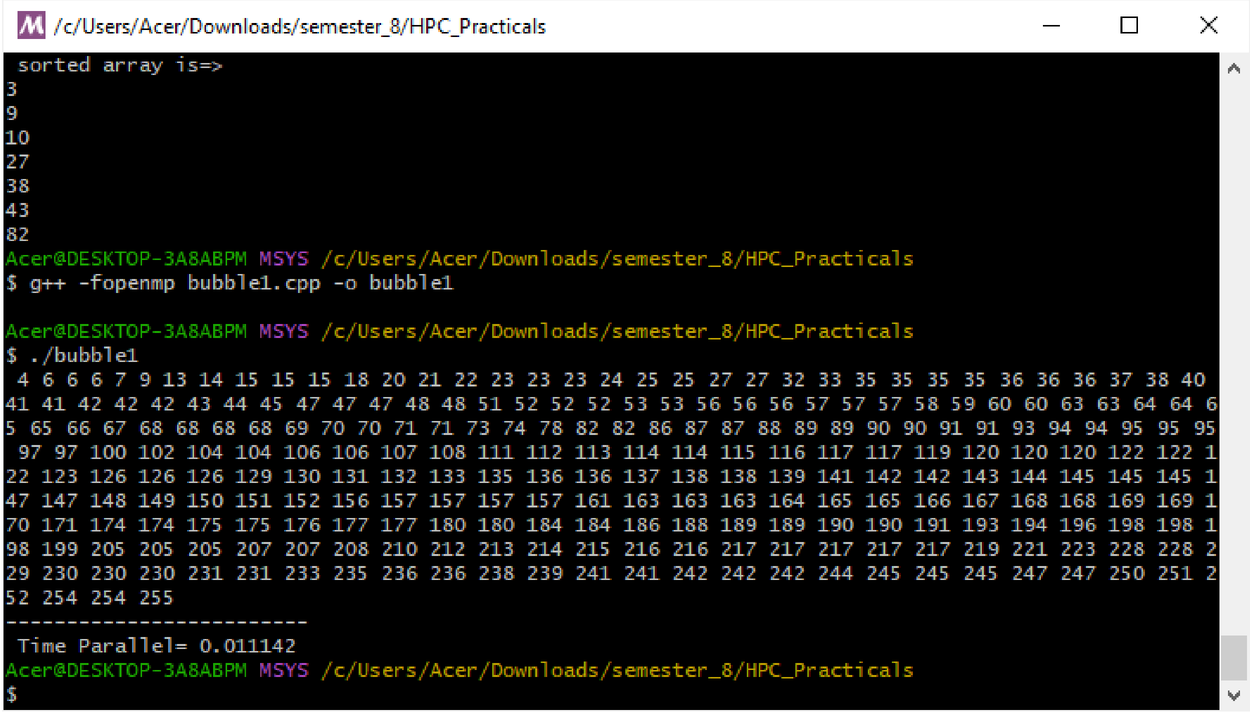
Merge Sort:

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


void mergesort(int a[],int i,int j);
void merge(int a[],int i1,int j1,int i2,int j2);

void mergesort(int a[],int i,int j)
{
    int mid;
    if(i<j)
    {
        mid=(i+j)/2;

        #pragma omp parallel sections
        {

            #pragma omp section
            {
                mergesort(a,i,mid);
            }

            #pragma omp section
            {
                mergesort(a,mid+1,j);
            }
        }

        merge(a,i,mid,mid+1,j);
    }

}

void merge(int a[],int i1,int j1,int i2,int j2)
{
    int temp[1000];
    int i,j,k;
    i=i1;
    j=i2;
    k=0;

    while(i<=j1 && j<=j2)
    {
        if(a[i]<a[j])
        {
            temp[k++]=a[i++];
        }
        else
        {
            temp[k++]=a[j++];
            }
    }

    while(i<=j1)
    {
        temp[k++]=a[i++];
    }

    while(j<=j2)
    {
        temp[k++]=a[j++];
    }
    for(i=i1,j=0;i<=j2;i++,j++)
    {
        a[i]=temp[j];
    }
}


int main()
{
    int *a,n,i;
    cout<<"\n enter total no of elements=>";
    cin>>n;
    a= new int[n];

    cout<<"\n enter elements=>\n";
    for(i=0;i<n;i++)
    {
        cin>>a[i];
    }

    mergesort(a, 0, n-1);

    cout<<"\n sorted array is=>";
    for(i=0;i<n;i++)
    {
        cout<<"\n"<<a[i];
    }

    return 0;
}
```

Output:
Bubble Sort:

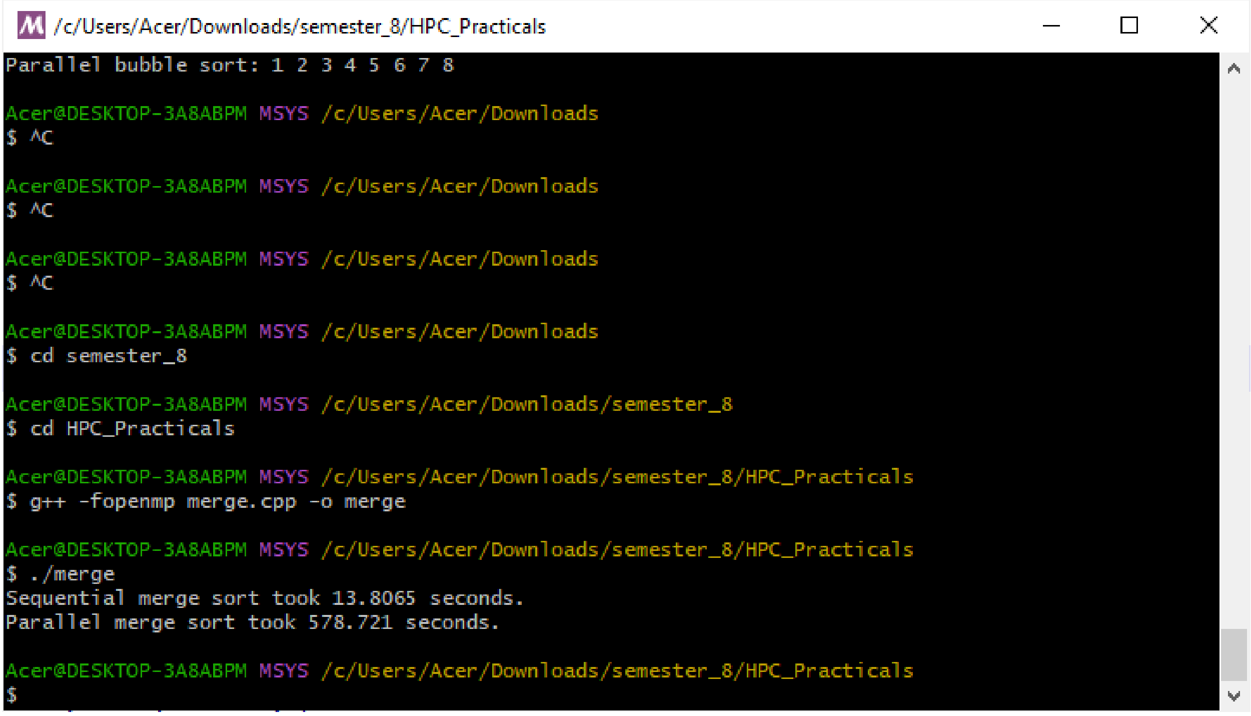


```
Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ./Bubble
-bash: ./Bubble: No such file or directory

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ g++ hello.cpp -o hello

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ./hello
Hello, world!

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ g++ -fopenmp Bubble.cpp -o Bubble

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ./Bubble
Segmentation fault (core dumped)

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ g++ -fopenmp hello.cpp -o hello

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ./hello
Sequential bubble sort: 1 2 3 4 5 6 7 8
Parallel bubble sort: 1 2 3 4 5 6 7 8

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
```

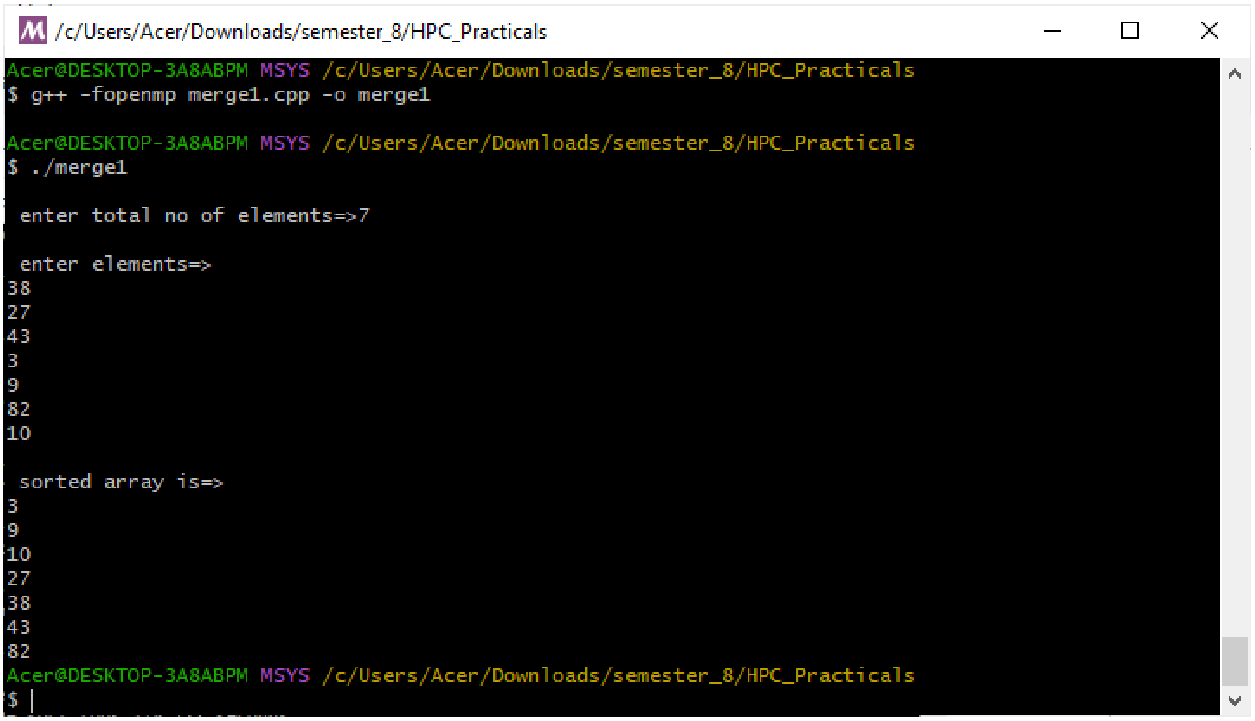


```
 sorted array is=>
3
9
10
27
38
43
82
Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ g++ -fopenmp bubble1.cpp -o bubble1

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ ./bubble1
 4 6 6 6 7 9 13 14 15 15 15 18 20 21 22 23 23 23 24 25 25 27 27 32 33 35 35 35 35 36 36 36 37 38 40
41 41 42 42 42 43 44 45 47 47 47 48 48 51 52 52 52 53 53 56 56 56 57 57 57 58 59 60 60 63 63 64 64 6
5 65 66 67 68 68 68 68 69 70 70 71 71 73 74 78 82 82 86 87 87 88 89 89 90 90 91 91 93 94 94 95 95 95
 97 97 100 102 104 104 106 106 107 108 111 112 113 114 114 115 116 117 117 119 120 120 120 122 122 1
22 123 126 126 126 129 130 131 132 133 135 136 136 137 138 138 139 141 142 142 143 144 145 145 145 1
47 147 148 149 150 151 152 156 157 157 157 157 161 163 163 163 164 165 165 166 167 168 168 169 169 1
70 171 174 174 175 175 176 177 177 180 180 184 184 186 188 189 189 190 190 191 193 194 196 198 198 1
98 199 205 205 205 207 207 208 210 212 213 214 215 216 216 217 217 217 217 217 219 221 223 228 228 2
29 230 230 230 231 231 233 235 236 236 238 239 241 241 242 242 242 244 245 245 245 247 247 250 251 2
52 254 254 255
-------------------------
 Time Parallel= 0.011142
Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$
```

Merge Sort:



```
Parallel bubble sort: 1 2 3 4 5 6 7 8

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ^C

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ^C

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ ^C

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads
$ cd semester_8

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8
$ cd HPC_Practicals

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ g++ -fopenmp merge.cpp -o merge

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ ./merge
Sequential merge sort took 13.8065 seconds.
Parallel merge sort took 578.721 seconds.

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$
```



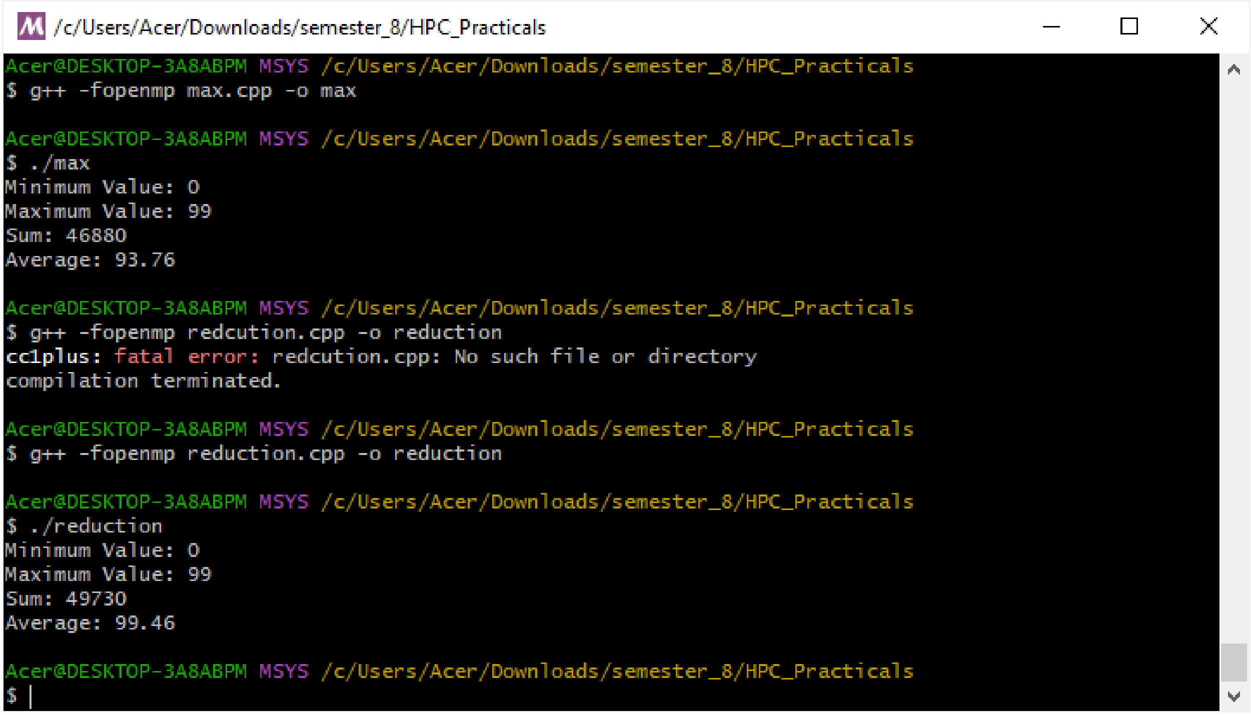
```
Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ g++ -fopenmp merge1.cpp -o merge1

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ ./merge1

 enter total no of elements=>7

 enter elements=>
38
27
43
3
9
82
10

 sorted array is=>
3
9
10
27
38
43
82
Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$
```

Assignment 3

Code:

```
#include <iostream>
#include <omp.h>
#include <cstdlib>
#include <ctime>

using namespace std;

int main() {
    int n = 1000;
    int arr[n];

    // Generate random array
    srand(time(NULL));
    for(int i=0;i<n;i++) {
        arr[i] = rand() % 100;
    }

            // Min Operation
    int min_val = arr[0];
    #pragma omp parallel for reduction(min:min_val)
    for(int i=1;i<n;i++) {
        if(arr[i] < min_val) {
            min_val = arr[i];
        }
    }
    cout << "Minimum Value: " << min_val << endl;


    // Max Operation
    int max_val = arr[0];
    #pragma omp parallel for reduction(max:max_val)
    for(int i=1;i<n;i++) {
        if(arr[i] > max_val) {
            max_val = arr[i];
        }
    }
    cout << "Maximum Value: " << max_val << endl;

    // Sum Operation
    int sum = 0;
    #pragma omp parallel for reduction(+:sum)
    for(int i=0;i<n;i++) {
        sum += arr[i];
    }
    cout << "Sum: " << sum << endl;

    // Average Operation
    double average = 0.0;
    #pragma omp parallel for reduction(+:sum)
    for(int i=0;i<n;i++) {
        sum += arr[i];
    }
    average = (double)sum / n;
    cout << "Average: " << average << endl;

    return 0;
}
```

Output:



```
Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ g++ -fopenmp max.cpp -o max

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ ./max
Minimum Value: 0
Maximum Value: 99
Sum: 46880
Average: 93.76

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ g++ -fopenmp redcution.cpp -o reduction
cc1plus: fatal error: redcution.cpp: No such file or directory
compilation terminated.

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ g++ -fopenmp reduction.cpp -o reduction

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ ./reduction
Minimum Value: 0
Maximum Value: 99
Sum: 49730
Average: 99.46

Acer@DESKTOP-3A8ABPM MSYS /c/Users/Acer/Downloads/semester_8/HPC_Practicals
$ |
```