LP-V

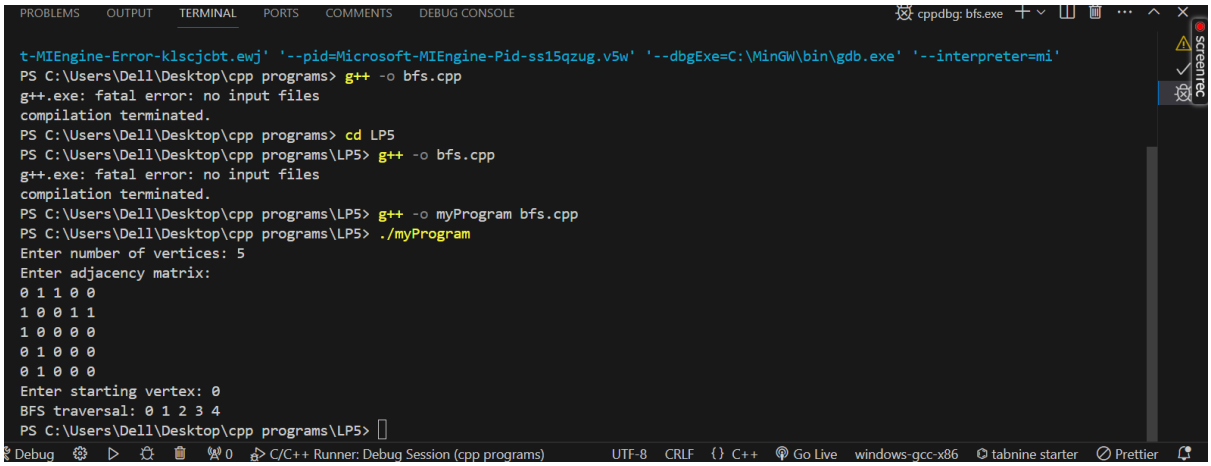
HPC Assignment No-1

BFS

```
#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;
}
```

OUTPUT

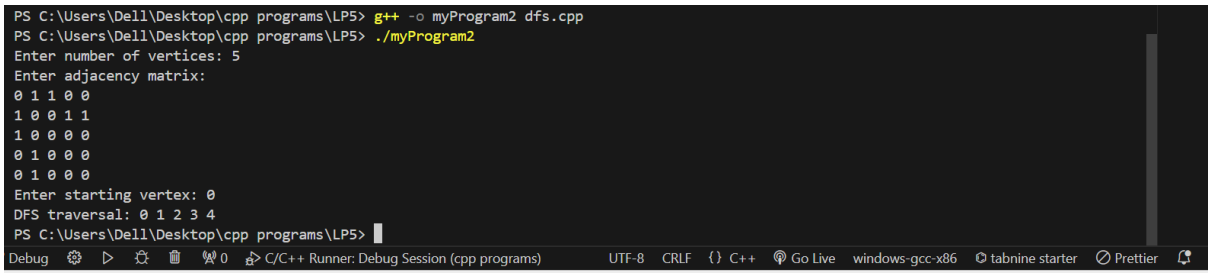


DFS

```
#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:



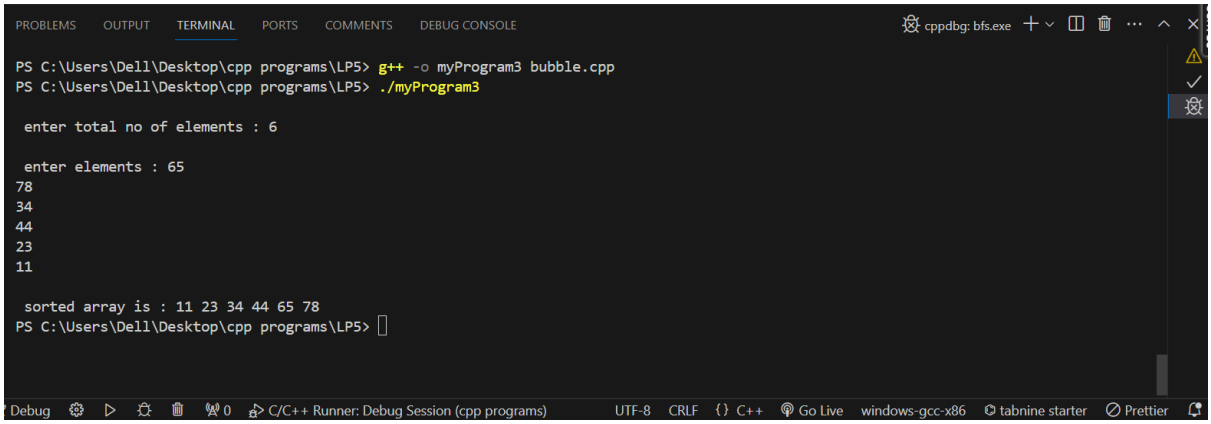
HPC Assignment No-2

Bubble Sort

```
#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;

        #pragma omp parallel for shared(a,first)
        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 test;
  test=a;
  a=b;
  b=test;
}
int main()
{
  int *a,n;
  cout<<"\n enter total no of elements=>";
  cin>>n;
  a=new int[n];

  cout<<"\n enter elements=>";

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

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


return 0;
}
```

Output :



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: