CS213 Final Project

Team Members-
Ravi Kumar Kushawaha- 150070045
Saqib Azim- 150070031
Prakhar Jain - 150040084
Deep Tavker - 150010001

node.cpp

```
#include <iostream>
#include <queue>
#include <deque>

class node{
public:
char[3] nodeID;
int num_wait;            //no. of people waiting
int slots[9];
std::deque<int> wait_line;
void student_arrival(int s);  //s->no. of students wait_lineived
int bus_arrival(Bus b);       //b-> bus_id
 void update_slot_data(int a);
 int calc_weight(void);

 //to define static members - routes
}


 int node::bus_arrival(Bus b)
{
        int vacant=b.current_capacity+b.get down;
        for (std::deque<int>::iterator it = wait_line.begin(); it !=
wait_line.end(); ++it)
        {if (*it<=vacant) {wait_line.pop(); vacant=vacant-*it;}
          else {*it=*it-vacant; vacant=0;}
```

```
        return vacant;
}

 void node::student_arrival(int s)
{
  slots[i]=slot[i]+j;
  for ((std::deque<int>::iterator it = wait_line.end()-1)<5)
wait_line.push_back(s);

  else{
  wait_line.pop();
  wait_line.push_back(s);}
}

void node::update_slot_data(int a)
{
        slot[i]=(slot[i]+a)/2;
}

int node:: calc_weight(void)
{ weight=0; i=0;
for (std::deque<int>::iterator it = wait_line.begin(); it != wait_line.end(); +
+it)
{
        weight=*it * i;
        i=i+2;
}
}
```

bus_lib.h

```
#include <iostream>
#include <cstring>
#include <string>
```

```
using namespace std;
#ifndef test

class Bus{
public:
	int busID;
	int bus_max_capacity;
	int bus_present_capacity;
	char source_node[3] = "000";
	char destination_node[3] = "000";
	int avg_velocity = 30;		// in km/hr
	bool active = false;
	char node_last_visited[3];
	int bus_stop_time;		// will depend on the number of people at that particular bus-stop
	string route[10];		// an array of a sequence of nodes

	// member functions declared
	// when the bus reaches a particular stop it should update the bus_capacity, bus_last_node and
	// will wait for unit_stop_time*no_of_people_at_that_stop

	void update_bus_parameters(char stop_node[]){
		node_last_visited = stop_node;

	}

	// this function will return the next node ID of the bus
	void get_next_node(node_last_visited, route){
		for(int i=0;i<length(route);i++){		// how to get the length of a string object
			if(strcmp(route[i], node_last_visited)){
				return route[i+1];
			}
		}
	}
```

```
        int get_distance(last_node, route){
                next_node = get_next_node(last_node, route);
                if(strcmp(last_node, "H12") && strcmp(next_node, "H07")){
                        return distance[0];   // problem here since we don't have
access to the distance array here
                }

        }

        void move_bus(){
                /*while(distance_travelled < next_node-bus_last_node){
                        distance_travelled += avg_velo
                }*/
                time_to_next_node = (get_distance(node_last_visited,
route)/avg_velocity)*scale;
                sleep(time_to_next_node);
        }

};
#endif
```

main.cpp

```
struct Interface{
        Bus bus_array[10];
        Node node_array[10];
        int distance[11];

        void update_distance(){
                cout<<"Update distances of adjacent stops"
                for(int i=0;i<11;i++){
                        cin>>distance[i];
                }
        }
```

```
};


int main(){
        Interface obj;
        obj.update_distance();   // will update the distances between the nodes

}
```

**** This code is not complete yet.**