Practical 5

18BCE243

Code of Practical 5

```
#ifndef GTLAB_PRACTICAL5
#define GTLAB_PRACTICAL5

#include <climits>
#include <iostream>
#include <vector>
#include <set>
#include <map>
#include <queue>

using namespace std;

typedef struct _vertex_t
{
    int v;
    struct _vertex_t *p;
} vertex_t;

vertex_t * djikstra(map<int, vector<pair<int, int> > > adj_list,
                    int init, int goal)
{
    priority_queue<pair<int, int> > frontier;   // currently exploring
    set<int> explored;                          // explored
    vertex_t *prev = NULL;

    // start with exploring the initial node
    frontier.push({0, init});

    // while there are nodes to explore, explore!
    while(!frontier.empty()) {
        auto curr = frontier.top();
        frontier.pop();
        vertex_t *new_vert = new vertex_t;
        new_vert->v = curr.second;
        new_vert->p = prev;
        prev = new_vert;

        // if goal is reached, return the path
        if (curr.second == goal) return new_vert;
        // if node is already explored, continue
```

```
        if (explored.find(curr.second) != explored.end()) continue;

        // add the current node to explored.
        explored.insert(curr.second);

        // visit all the neighbors and start exploring.
        for (auto &i: adj_list[curr.second]) {
            frontier.push({curr.first + i.first, i.second});
        }
    }

    // failed
    return NULL;
}

#endif // GTLAB_PRACTICAL5
```

Test Driver (with Inputs)

```
#include "practical5.h"

int main()
{
    map<int, vector<pair<int, int> >> adj_list;
    adj_list[1] = {{5, 2}};
    adj_list[2] = {{2, 3}};
    adj_list[3] = {{4, 4}};
    adj_list[4] = {{1, 5}};
    int init = 1, goal = 5;

    vertex_t *path = djikstra(adj_list, init, goal);
    while(path) {
        cout << path->v << " ";
        path = path->p;
    }
    cout << endl;
    return 0;
}
```

Output

```
5 4 3 2 1
```