Course: Data Structures & Algorithm

Homework 5 **Due date: 12 June, 2014 (before the class)**

This is the Individual Homework. One who copies other work or whose work is being copied by others will receive the worst final grade F, regardless of any excuse.

There is a pictured graph for the intimacy relationship in a community of people (see the attached file in YSCEC), where each node denotes a member and the number in the edge represents the intimacy level between a pair of members. The higher number indicates closer intimacy between a pair of members. For this homework, we will surf the human network of 37 members.

(1) Write a program to build the adjacency matrix by reading an input file to encode the graph information (use `fscanf` command). The input file 'member.txt' is given in the following format:

member No.	member No.	intimacy level
7	8	5
7	17	5
8	9	5

…

There is no self-loop for each node, but even isolated nodes should be defined in the program.

(2) Write a program to build the adjacency matrix using an array.

(3) Write a program to build the adjacency list using the linked list structure. You can read the input file to build it or convert the adjacency matrix into the adjacency list. Define the data structure appropriately to obtain the adjacency list.

(4) Write a program to find who has the largest number of friends among the members in the group. Show the result using the adjacency matrix and the adjacency list, respectively.

(5) Write a depth-first-search program (recursive program) *dfs* in the lecture note to find *connected components* of the graph. Show how many subgroups of the human network in the community exist, using the adjacency matrix and the adjacency list, respectively. For each connected component, show the number of members which belong to the same connected component.
You need to show two results with adjacency matrix and adjacency list. The member order will be different with the two results.

(6) Repeat the above problem (5) with the breadth-first search. For each connected component, show the number of members which belong to the same connected component. You need to show two results with adjacency matrix and adjacency list. Here, you need to show the content of queue just before every de-queue operation and the selected node from the queue (popped element from the queue).

(7) The following program is a stack-based search method. This is the same algorithm as the breadth first search method, but only changes the enqueue() by push() and the dequeue() by pop(). Repeat the problem (5) with the stack-based search method. For each connected component, show the number of members which belong to the same connected component. You need to show two results with adjacency matrix and adjacency list.
Here, you need to show the content of stack just before every pop operation and the selected node from the stack (popped element from the stack).

```
int stack[100];
top= -1;

void search(int v)
{
   nodepointer w;
   top= -1; printf("%5d", v);
   visited[v] = TRUE;
   push(v);
  for (top >=0) {
       // show the stack content before pop() operation
       v=pop();
       // show the popped element v
       for(w=adjList[v]; w != NULL; w = w->link)
       if (!visited[w->vertex]) {
            printf("%3d ", w->vertex);
            push(w->vertex);
            visited[w->vertex] = TRUE; }
                                }
                 }
}
```

You need to write a program for each task. All the detailed works described above should be completed with programs. .

Analyze the time complexity for each task. You need to submit a report including documents as well.