BFS and DFS

```
/* Roll no. : 2002
Batch : E-10
*/

#include <iostream>
#include "adj_list.h"
#include<stdlib.h>
using namespace std;

int main()
{
	adj_list A;

	do
	{
		int ch;
		cout<<"\tMENU\n";

		cout<<"\t\t1.Create\n\t\t2.Display\n\t\t3.Friends\n\t\t4.Comments"
			"\n\t\t5.Date of birth\n\t\t6.Exit\n\n";
		cout<<"Enter Choice : ";
		cin>>ch;
		cout<<endl;

		switch(ch)
		{
			case 1:
				{
					int n;

					cout<<"Enter number of users in your network : ";
					cin>>n;
					cout<<endl;

					A.root=A.create(n);
					break;
				}

			case 2:
			{
					A.display(A.root);
					break;
			}

			case 3:
			{
					A.bfs(A.root);
					A.unvisit();
					A.dfs(A.root,0);
```

```
                                        cout<<"User having maximum friends : "<<A.popular<<endl;
                                        cout<<"Number of "<<A.popular<<" has is :
"<<A.max_friends<<endl;
                                        break;

                        }

                        case 4:
                        {
                                        A.unvisit();
                                        A.dfs(A.root,0);

                                        cout<<"Maximum comments have been made by :
"<<A.social<<endl;
                                        cout<<"Number of comments made by "<<A.social<<" is
"<<A.max<<endl<<endl;

                                        cout<<"Minimum comments have been made by :
"<<A.introvert<<endl;
                                        cout<<"Number of comments made by "<<A.introvert<<" is
"<<A.min<<endl;

                                        break;
                        }

                        case 5:
                        {
                                        int month;
                                        cout<<"Enter month you are intrested in  : ";
                                        cin>>month;
                                        cout<<endl;

                                        A.unvisit();
                                        A.dfs(A.root,month);
                                        cout<<"Number of people having date of birth in month "
                                        <<month<<" : "<<A.same_month<<endl;

                                        break;
                        }

                case 6:         exit(0);

                }
        }
        while(1);

        return 0;
}
```

adj_list.h

```
/*
 * adj_list.h
 *
 *  Created on: 21-Feb-2018
 *      Author: e2002
 */
#include<iostream>
#include<string>
using namespace std;


struct frnd
{
	string name;
	frnd * next;

};

struct node
{
	string Name;
	int dd,mm,yy;
	int post;
	node * down;
	frnd * Next;
	int visited;
	int friends;
};

#ifndef ADJ_LIST_H_
#define ADJ_LIST_H_

class adj_list
{

public:

		node * root;

		int max;
		int min;
		int same_month;
		int max_friends;
		string popular;
		string social;
		string introvert;

		adj_list();
		node * create(int);
```

```
                frnd * add_friend(node *);
                void display(node *);

                void dfs(node *,int);
                void bfs(node *);
                node * search(string);
                void unvisit();
};

#endif /* ADJ_LIST_H_ */
```

adj_list.cpp

```
/*
 * adj_list.cpp
 *
 *  Created on: 21-Feb-2018
 *      Author: e2002
 */

#include "adj_list.h"
#include<iostream>
#include<string>
#include<malloc.h>
#include<queue>
using namespace std;

adj_list :: adj_list()
{
	max=0;
	min=9999;
	same_month=0;
	max_friends=0;
}

void getdata(node * p)
{
	cout<<"Enter Name : ";
	cin>>p->Name;
	cout<<endl;

	cout<<"Enter Date of Birth(DD/MM/YYYY) : ";
	cin>>p->dd;
	cin>>p->mm;
	cin>>p->yy;
	cout<<endl;

	cout<<"Enter post : ";
	cin>>p->post;
	cout<<endl;

	p->visited=0;
	p->friends=0;

}

frnd * adj_list:: add_friend(node * root)
{
	frnd *head,*p,*q;

		head = new frnd;
```

```
		cout<<"Enter Name of 1st friend of "<<root->Name<<" : ";
		cin>>head->name;
		cout<<endl;

		head->next=NULL;

		p=head;
		int i=1;

		cout<<"More friends?(Y/N) of "<<root->Name<<" : ";
		char ch;
		cin>>ch;
		cout<<endl;

		if(ch=='N' || ch=='n')
			return head;


		do
		{
			q=new frnd;

			cout<<"Enter name of frind  no. "<<i+1<<" : ";
			cin>>q->name;
			cout<<endl;

			q->next=NULL;

			p->next=q;
			p=q;

			i++;

			cout<<"More friends?(Y/N) of "<<root->Name<<" : ";
			char ch;
			cin>>ch;
			cout<<endl;

			if(ch=='N' || ch=='n')
				break;

		}
		while(1);

		return head;
}

node * adj_list:: create(int n)
{

	node *head,*p,*q;
	head = new node;
```

```
	cout<<"Enter Data of 1st Person : ";
	cout<<endl<<endl;
	getdata(head);
	cout<<endl;

	head->down=NULL;
	head->Next=NULL;

	p=head;

	for(int i=1;i<n;i++)
	{
		q=new node;

		cout<<"Enter data of person no. "<<i+1<<" : ";
		cout<<endl<<endl;
		getdata(q);
		cout<<endl;

		q->down=NULL;
		q->Next=NULL;

		p->down=q;
		p=q;

	}

	p=head;

	while(p != NULL)
	{
		p->Next=add_friend(p);
		p=p->down;
	}


	return head;
}



void adj_list :: display(node *p)
{
	node *head;

	head=p;

	while(head != NULL)
	{
		cout<<""<<head->Name;
```

```
                frnd  *q;
                        q=head->Next;

                        while(q != NULL)
                        {
                                cout<<"<- "<<q->name;
                                q=q->next;
                        }

                cout<<endl;
                head=head->down;
        }

        cout<<"NULL";
        cout<<endl;
}

void adj_list :: unvisit()
{
        node *p;
        p=root;

        while(p != NULL)
        {
                p->visited=0;
                p=p->down;
        }
}

node * adj_list :: search(string val)
{
        if(root->Name == val)
                return root;

        node * p;
        p=root;

        while(p != NULL && p->Name != val)
                p=p->down;;

        if(p->Name == val)
                return p;

        else
                return NULL;
}

void adj_list ::  dfs(node * root,int month)
{
        if(root->visited == 1)
                return ;
```

```
	root->visited = 1;

	if(root->post < min)
	{
		min=root->post;
		introvert=root->Name;
	}

	if(root->post > max)
	{
		max=root->post;
		social=root->Name;
	}

	if(root->mm == month)
		same_month++;

	if(root->friends > max_friends)
		{
			max_friends=root->friends;
			popular=root->Name;
		}

	frnd * p;
	p=root->Next;

	while(p != NULL)
	{
		node * temp;
		temp=search(p->name);
		dfs(temp,month);

		p=p->next;
	}
}

void adj_list :: bfs(node * current)
{
	queue<node *>q;

	q.push(current);
	current->visited=1;

	while(! q.empty())
	{
		frnd * p;
		p = q.front()->Next;

		while(p != NULL)
		{
			node * temp;
			temp=search(p->name);
```

```
                    if(temp->visited == 0)
                    {
                            temp->visited=1;
                            q.push(temp);
                    }

                    q.front()->friends++;
                    p=p->next;
            }
            q.pop();
    }
}
```