1. Write a program to insert given keys in hash table

2. Write a program to represent graph using adjacency matrix

```
#include<iostream>
using namespace std;
int vertArr[20][20]; //the adjacency matrix initially 0
int count = 0;
void displayMatrix(int v) {
  int i, j;
  for(i = 0; i < v; i++) {
    for(j = 0; j < v; j++) {
      cout << vertArr[i][j] << " ";
    }
    cout << endl;
  }
}
void add_edge(int u, int v) { //function to add edge into the matrix
  vertArr[u][v] = 1;
  vertArr[v][u] = 1;
}
main(int argc, char* argv[]) {
int v = 6; //there are 6 vertices in the graph
add_edge(0, 4);
add_edge(0, 3);
add_edge(1, 2);
add_edge(1, 4);
add_edge(1, 5);
add_edge(2, 3);
add_edge(2, 5);
add_edge(5, 3);
add_edge(5, 4);
displayMatrix(v);

}
```

3. Write C program to check entered graph is connected or not

```
#include <iostream>
using namespace std;
int main()
{
    int vertices;
   static int count;
```

```
    cout<<" Enter number of vertices ";
    cin>>vertices;
    int value;
    int adj [vertices] [vertices];
    for(int i=1;i<=vertices;i++){
        for(int j=1;j<=vertices;j++){
            adj[i][j]=0;
        }
    }
    cout<<" Inter THe values: ";
    for(int i=1;i<=vertices;i++){
        for(int j=1;j<=vertices;j++){
            cout<<i<<"\t"<<j;
            cin>>value;
            adj[i][j]=value;

        }
    }

        for(int i=1;i<=vertices;i++){
        for(int j=1;j<=vertices;j++){
            cout<<adj[i][j]<<"\t";

        }
        cout<<"\n";
    }

    for(int i=1;i<=vertices;i++){
         count=0;
        for(int j=1;j<=vertices;j++){

           if(adj[i][j]==0){
               count++;

           }
```

```
            else{
                count=0;
            }
        }
                if(count==vertices)
                {
                    cout<<"this is not connected graph";
                    break;
                }
        }

    return 0;
}
```

4. Write a program to insert key in hash table with chaining

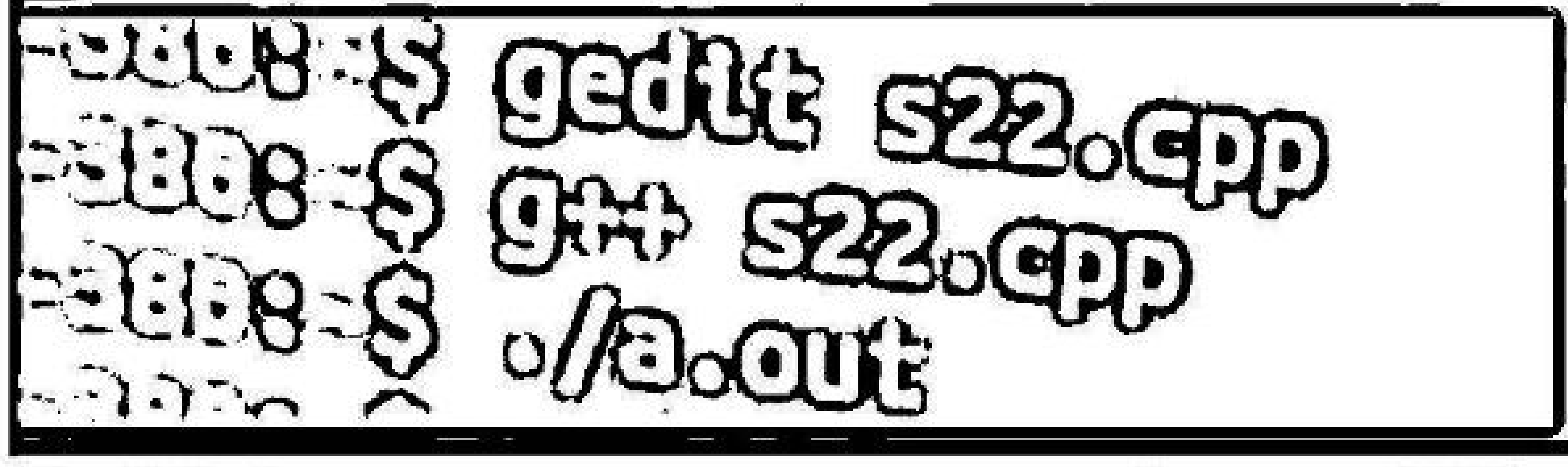
```
// CPP program to implement hashing with chaining
#include<bits/stdc++.h>
using namespace std;

class Hash
{
	int BUCKET; // No. of buckets

	// Pointer to an array containing buckets
	list<int> *table;
public:
	Hash(int V); // Constructor

	// inserts a key into hash table
	void insertItem(int x);

	// deletes a key from hash table
	void deleteItem(int key);

	// hash function to map values to key
```

```
        int hashFunction(int x) {
                return (x % BUCKET);
        }

        void displayHash();
};

Hash::Hash(int b)
{
        this->BUCKET = b;
        table = new list<int>[BUCKET];
}

void Hash::insertItem(int key)
{
        int index = hashFunction(key);
        table[index].push_back(key);
}

void Hash::deleteItem(int key)
{
// get the hash index of key
int index = hashFunction(key);

// find the key in (index)th list
list <int> :: iterator i;
for (i = table[index].begin();
                i != table[index].end(); i++) {
        if (*i == key)
        break;
}

// if key is found in hash table, remove it
if (i != table[index].end())
        table[index].erase(i);
```

```
}

// function to display hash table
void Hash::displayHash() {
for (int i = 0; i < BUCKET; i++) {
        cout << i;
        for (auto x : table[i])
        cout << " --> " << x;
        cout << endl;
}
}

// Driver program
int main()
{
// array that contains keys to be mapped
int a[] = {15, 11, 27, 8, 12};
int n = sizeof(a)/sizeof(a[0]);

// insert the keys into the hash table
Hash h(7); // 7 is count of buckets in
                    // hash table
for (int i = 0; i < n; i++)
        h.insertItem(a[i]);

// delete 12 from hash table
h.deleteItem(12);

// display the Hash table
h.displayHash();

return 0;
}
```

5. Write a program to insert data in file and write data to a file

```cpp
#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    ofstream myfile;
    myfile.open("pfile.txt");
    myfile<<"this is sample text";
    myfile<<"have a nice day ";
    myfile.close();
    return 0;
}
```



```
-388:~$ gedit s22.cpp
-388:~$ g++ s22.cpp
-388:~$ ./a.out
```

6. Write a program to perform different operations on set (with Python)

```python
# Program to perform different set operations like in
mathematics

# define three sets
E = {0, 2, 4, 6, 8};
N = {1, 2, 3, 4, 5};

# set union
print("Union of E and N is",E | N)

# set intersection
print("Intersection of E and N is",E & N)

# set difference
```

```
print("Difference of E and N is",E - N)

# set symmetric difference
print("Symmetric difference of E and N is",E ^ N)
```