Sanjiv Kannaa J
106121116

DBMS LAB – 8

Q1 Write a C program to find candidate key from functional dependancies

```
#include<bits/stdc++.h>
using namespace std;

int smallest_size = INT_MAX;

int obtain_bitmask(string A, unordered_map<char,int>mapping){
    int curr = 0;
    for(auto x: A){
        if(mapping.find(x)!= mapping.end()){
            curr |= 1<<(mapping[x]);
        }
    }
    return curr;
}

string convertToString(int mask, unordered_map<int,char>revMap){
    string res;
    int index = 0;
    while(mask>0){
        if(mask %2){
            res += revMap[index];
        }
        index++;
        mask /=2;
    }
    return res;
}
```

```
bool isSuperKey(string A,int set,
unordered_map<char,int>mapping,unordered_map<string,string>
func_depend, unordered_map<int,int> bit_depend){
    unordered_set<int>characters;
    int curr_set = set;
    while(true){
        int prev_set = curr_set;
        for(auto x: bit_depend){
            if((curr_set & x.first) != 0){
                curr_set |= x.second;
            }
        }
        if(curr_set == prev_set){
            break;
        }
    }
    if(curr_set == ((1<<A.size())-1)){
        return true;
    }
    return false;
}
int main(){
    int n;
    string A;
    unordered_map<char,int>mapping;
    unordered_map<int,char>revChar;
    unordered_map<string,string> func_depend;
    unordered_map<int,int> bit_depend;
    cout<<"Enter Attributes ";
    cin>>A;
    cout<<"Enter number of functional dependencies ";
    cin>>n;
    for(int i=0;i<A.size();i++){
        mapping[A[i]] = i;
```

```
        revChar[i] =A[i];
    }
    for(int i=0;i<n;i++){
        string LHS,RHS;
        cout<<"Enter the LHS of the string ";
        cin>>LHS;
        cout<<"Enter the RHS of the string ";
        cin>>RHS;
        int templ = obtain_bitmask(LHS,mapping), tempr =
obtain_bitmask(RHS,mapping);
        cout<<templ<<" "<<tempr<<endl;
        func_depend[LHS] = RHS;
        bit_depend[templ] = tempr;
    }

    for(int i=0;i<(1<<A.size());i++){
        if(isSuperKey(A,i,mapping, func_depend,bit_depend)){
            string temp= convertToString(i,revChar);
            if(temp.size() <=smallest_size){
                cout<<temp<<" is a candidate key"<<endl;
                smallest_size = temp.size();
            }
        }
    }
}
```

Q2 Write a C program to find super keys from functional dependancies

```
#include<bits/stdc++.h>
using namespace std;

int obtain_bitmask(string A, unordered_map<char,int>mapping){
    int curr = 0;
    for(auto x: A){
        if(mapping.find(x)!= mapping.end()){
            curr |= 1<<(mapping[x]);
        }
    }
    return curr;
}

string convertToString(int mask, unordered_map<int,char>revMap){
    string res;
    int index = 0;
    while(mask>0){
        if(mask %2){
            res += revMap[index];
        }
        index++;
        mask /=2;
    }
    return res;
}

bool isSuperKey(string A,int set,
unordered_map<char,int>mapping,unordered_map<string,string>
func_depend, unordered_map<int,int> bit_depend){
    unordered_set<int>characters;
    int curr_set = set;
    while(true){
```

```
        int prev_set = curr_set;
        for(auto x: bit_depend){
            if((curr_set & x.first) != 0){
                curr_set |= x.second;
            }
        }
        if(curr_set == prev_set){
            break;
        }
    }
    if(curr_set == ((1<<A.size())-1)){
        return true;
    }
    return false;
}
int main(){
    int n;
    string A;
    unordered_map<char,int>mapping;
    unordered_map<int,char>revChar;
    unordered_map<string,string> func_depend;
    unordered_map<int,int> bit_depend;
    cout<<"Enter Attributes ";
    cin>>A;
    cout<<"Enter number of functional dependencies ";
    cin>>n;
    for(int i=0;i<A.size();i++){
        mapping[A[i]] = i;
        revChar[i] =A[i];
    }
    for(int i=0;i<n;i++){
        string LHS,RHS;
        cout<<"Enter the LHS of the string ";
        cin>>LHS;
        cout<<"Enter the RHS of the string ";
```

```
        cin>>RHS;
        int templ = obtain_bitmask(LHS,mapping), tempr =
obtain_bitmask(RHS,mapping);
        cout<<templ<<" "<<tempr<<endl;
        func_depend[LHS] = RHS;
        bit_depend[templ] = tempr;
    }

    for(int i=0;i<(1<<A.size());i++){
        if(isSuperKey(A,i,mapping, func_depend,bit_depend)){
            cout<<convertToString(i,revChar)<<" is a super key"<<endl;
        }
    }
}
```