CODE

#*include*<*bits/stdc++.h*>

*using* *namespace* std;

#*define* end \_end

#*define* next \_nxt

*int* total=1000;

*const* *int* Max=500500;

*int* end[Max];

*int* next[27][Max];

*int* SC\_count=0;

*bool* created[Max];

*int* sz=0;

vector<string> AC\_ans;

vector<string> S\_ans;

*void* FTS\_insert(*string &s*)

{

*int* v=0;

*for*(*int* i=0;i<*s*.size();i++)

    {

*int* c=*s[*i*]-*'*a*';

*if*(!created[next[c][v]])

        {

            next[c][v]=++sz;

            created[sz]=1;

        }

        v=next[c][v];

    }

    ++end[v];

}

*bool* FTS\_search(*string tmp*)

{

*int* v=0;

*for*(*int* i=0;i<*tmp*.size();i++)

    {

*int* c=*tmp[*i*]-*'*a*';

*if*(!created[next[c][v]])

*return* false;

        v=next[c][v];

    }

*return* end[v]>0;

}

*int* Full\_text\_search()

{

    ifstream myFile;

    myFile.open("*test.csv*");

*int* row\_count =0 ;

    string line;

*int* i=0;

    string key[total];

*while*(getline(myFile,line))

    {

        row\_count +=1;

        stringstream  lineStream(line);

        string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

        {

            column\_count+=1;

*if* ( column\_count == 2)

            {

                key[i]*=*cell;

                i++;

                FTS\_insert(cell);

            }

        }

    }

    myFile.close();

    string a;

    cout*<<*"*\n INSERT THE NAME YOU WANT TO FIND IN CONTACT LIST :*"*<<*endl;

    cin*>>*a;

    cout*<<*endl;

*if*(FTS\_search(a)==0)

       cout*<<*"*Record not found in your contact list.*"*<<*endl;

*if*(FTS\_search(a))

    {

        cout*<<*"*Record found in your contact list.*"*<<*endl;

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*int* pos=0;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if* ( column\_count == 2)

                {

*if*(cell.compare(a)==0)

                    {

                        pos=row\_count;

*break*;

                    }

                }

            }

        }

        myFile.close();

        myFile.open("*test.csv*");

        row\_count =0 ;

        string line1;

*while*(getline(myFile,line1))

        {

            row\_count +=1;

            stringstream  lineStream(line1);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*(row\_count==pos && (column\_count==1 || column\_count==2 || column\_count==3 || column\_count==4 || column\_count==5))

                {

*if*(column\_count==1)

                        cout*<<*"*Id \t\t:*";

*if*(column\_count==2)

                        cout*<<*"*Name \t\t:*";

*if*(column\_count==3)

                        cout*<<*"*Contact Number  :*";

*if*(column\_count==4)

                        cout*<<*"*Operator \t:*";

*if*(column\_count==5)

                        cout*<<*"*Location \t:*";

                    cout*<<*cell*<<*endl;

                }

            }

        }

    }

*return* 0;

}

*struct* S\_Trie

{

*int* index;

    S\_Trie*\** child[26];

    S\_Trie()

    {

*for* (*int* i = 0; i < 26; i++)

            child[i] = NULL;

        index = *-*1;

    }

};

*void* S\_insert(*S\_Trie\* root*,*string str*,*int index*)

{

    S\_Trie*\** node = *root*;

*for* (*int* i = 0; i < *str*.size(); i++) {

*char* ind = *str[*i*]* *-* '*a*';

*if* (!node->child[ind])

            node->child[ind] = *new* S\_Trie();

        node = node->child[ind];

    }

    node->index = *index*;

}

*bool* S\_preorder(*S\_Trie\* node*,*string arr*[])

{

*if* (*node* == NULL)

*return* false;

*for* (*int* i = 0; i < 26; i++)

    {

*if* (*node*->child[i] != NULL)

        {

*if* (*node*->child[i]->index != *-*1)

                S\_ans.push\_back(*arr*[*node*->child[i]->index]);

            S\_preorder(*node*->child[i], *arr*);

        }

    }

}

*void* S\_printSorted(*string arr*[],*int n*)

{

    S\_Trie*\** root = *new* S\_Trie();

*for* (*int* i = 0; i < *n*; i++)

        S\_insert(root, *arr*[i], i);

    S\_preorder(root, *arr*);

}

*int* sort\_name()

{

*int* a,b;

    cout*<<*"*\n ENTER THE RANGE (A,B) :*"*<<*endl;

    cin*>>*a*>>*b;

    cout*<<*endl;

    cout*<<*"*\t\t\t\t\t\tContact list sorted by Name :\n*"*<<*endl;

    cout*<<*"*ID\t\t\tNAME\t\t\tNUMBER\t\t\tOPERATOR\t\tLOCATION*"*<<*endl;

*int* n;

    n=b*-*a*+*1;

*int* i=0;

    string arr[n];

    ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*(column\_count==2 *and* row\_count>=a *and* row\_count<=b)

                {

                    arr[i]*=*cell;

                    i++;

*break*;

                }

            }

        }

    S\_printSorted(arr, n);

*int* q=S\_ans.size();

*int* index[q]={*-*1};

*int* p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*int* vai=p;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

                string abc=S\_ans.at(p);

*if* ( column\_count == 2)

                {

*if*(cell.compare(abc)==0)

                    {

                        index[p]=row\_count;

                        p++;

*break*;

                    }

                }

            }

*if*(vai<p)

            {

*break*;

            }

        }

    }

    q=S\_ans.size();

    p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*((index[p]==row\_count) && (column\_count==1 || column\_count==2 || column\_count==3 || column\_count==4 || column\_count==5))

                {

*if*(cell.size()<8)

                        cout*<<*cell*<<*"*\t\t\t*";

*else*

                        cout*<<*cell*<<*"*\t\t*";

                }

            }

*if*(index[p]==row\_count)

            {

                p++;

                cout*<<*endl;

            }

        }

    }

    S\_ans.clear();

*return* 0;

}

*int* sort\_operator()

{

*int* a,b;

    cout*<<*"*\n ENTER THE RANGE (A,B) :*"*<<*endl;

    cin*>>*a*>>*b;

    cout*<<*endl;

    cout*<<*"*INPUT THE OPERATOR :*"*<<*endl;

    string opr;

    cin*>>*opr;

    cout*<<*"*\t\t\t\t\t\tContact list sorted by Operator :\n*"*<<*endl;

    cout*<<*"*ID\t\t\tNAME\t\t\tNUMBER\t\t\tOPERATOR\t\tLOCATION*"*<<*endl;

*int* n;

    n=b*-*a*+*1;

*int* i=0;

    string arr[n];

    ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*(column\_count==2 *and* row\_count>=a *and* row\_count<=b)

                {

                    arr[i]*=*cell;

                }

*if*(column\_count==4 *and* row\_count>=a *and* row\_count<=b)

                {

*if*(cell.compare(opr)==0)

                    {

                        i++;

*break*;

                    }

                }

            }

        }

    S\_printSorted(arr, i);

*int* q=S\_ans.size();

*int* index[q]={*-*1};

*int* p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*int* vai=p;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

                string abc=S\_ans.at(p);

*if* ( column\_count == 2)

                {

*if*(cell.compare(abc)==0)

                    {

                        index[p]=row\_count;

                        p++;

*break*;

                    }

                }

            }

*if*(vai<p)

            {

*break*;

            }

        }

    }

    q=S\_ans.size();

    p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*((index[p]==row\_count) && (column\_count==1 || column\_count==2 || column\_count==3 || column\_count==4 || column\_count==5))

                {

*if*(cell.size()<8)

                        cout*<<*cell*<<*"*\t\t\t*";

*else*

                        cout*<<*cell*<<*"*\t\t*";

                }

            }

*if*(index[p]==row\_count)

            {

                p++;

                cout*<<*endl;

            }

        }

    }

    S\_ans.clear();

*return* 0;

}

*int* sort\_loc()

{

*int* a,b;

    cout*<<*"*\n ENTER THE RANGE (A,B) :*"*<<*endl;

    cin*>>*a*>>*b;

    cout*<<*endl;

    cout*<<*"*INPUT THE LOCATION :*"*<<*endl;

    string loc;

    cin*>>*loc;

    cout*<<*"*\t\t\t\t\t\tContact list sorted by Location :\n*"*<<*endl;

    cout*<<*"*ID\t\t\tNAME\t\t\tNUMBER\t\t\tOPERATOR\t\tLOCATION*"*<<*endl;

*int* n;

    n=b*-*a*+*1;

*int* i=0;

    string arr[n];

    ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*(column\_count==2 *and* row\_count>=a *and* row\_count<=b)

                {

                    arr[i]*=*cell;

                }

*if*(column\_count==5 *and* row\_count>=a *and* row\_count<=b)

                {

*if*(cell.compare(loc)==0)

                    {

                        i++;

*break*;

                    }

                }

            }

        }

    S\_printSorted(arr, i);

*int* q=S\_ans.size();

*int* index[q]={*-*1};

*int* p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*int* vai=p;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

                string abc=S\_ans.at(p);

*if* ( column\_count == 2)

                {

*if*(cell.compare(abc)==0)

                    {

                        index[p]=row\_count;

                        p++;

*break*;

                    }

                }

            }

*if*(vai<p)

            {

*break*;

            }

        }

    }

    q=S\_ans.size();

    p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*((index[p]==row\_count) && (column\_count==1 || column\_count==2 || column\_count==3 || column\_count==4 || column\_count==5))

                {

*if*(cell.size()<8)

                        cout*<<*cell*<<*"*\t\t\t*";

*else*

                        cout*<<*cell*<<*"*\t\t*";

                }

            }

*if*(index[p]==row\_count)

            {

                p++;

                cout*<<*endl;

            }

        }

    }

    S\_ans.clear();

*return* 0;

}

*int* sort\_op()

{

*int* choice;

    cout*<<*"*\n\n\n\t\t\t\t\t\t  SORT OPERATIONS in a given range (a,b)*"*<<*endl;

    cout*<<*endl*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 1: SORT CONTACT LIST BY NAME*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 2: SORT CONTACT LIST BY OPERATOR*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 3: SORT CONTACT LIST BY LOCATION*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 4: EXIT*"*<<*endl*<<*endl;

    cout*<<*"*------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------*"*<<*endl;

    cin*>>*choice;

*switch*(choice)

    {

*case* 1:

        {

            system("*cls*");

            sort\_name();

*break*;

        }

*case* 2:

        {

            system("*cls*");

            sort\_operator();

*break*;

        }

*case* 3:

        {

            system("*cls*");

            sort\_loc();

*break*;

        }

*case* 4:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\t\t\t\t\t\tThank you*";

*break*;

        }

*default*:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\n \t\t\t\t\t\tWrong choice entered*"*<<*endl;

*break*;

        }

    }

}

*struct* TrieNode\_AC

{

*struct* TrieNode\_AC *\**children[26];

*bool* isWordEnd;

};

*struct* TrieNode\_AC *\**getNode(*void*)

{

*struct* TrieNode\_AC *\**pNode = *new* TrieNode\_AC;

    pNode->isWordEnd = false;

*for* (*int* i = 0; i < 26; i++)

        pNode->children[i] = NULL;

*return* pNode;

}

*void* AC\_insert(*struct TrieNode\_AC \*root*,*const string key*)

{

*struct* TrieNode\_AC *\**pCrawl = *root*;

*for* (*int* level = 0; level < *key*.length(); level++)

    {

*int* index = *key[*level*]-*'*a*';

*if* (!pCrawl->children[index])

            pCrawl->children[index] = getNode();

        pCrawl = pCrawl->children[index];

    }

    pCrawl->isWordEnd = true;

}

*bool* AC\_search(*struct TrieNode\_AC \*root*,*const string key*)

{

*int* length = *key*.length();

*struct* TrieNode\_AC *\**pCrawl = *root*;

*for* (*int* level = 0; level < length; level++)

    {

*int* index = *key[*level*]-*'*a*';

*if* (!pCrawl->children[index])

*return* false;

        pCrawl = pCrawl->children[index];

    }

*return* (pCrawl != NULL && pCrawl->isWordEnd);

}

*bool* AC\_isLastNode(*struct TrieNode\_AC \*root*)

{

*for* (*int* i = 0; i < 26; i++)

*if* (*root*->children[i])

*return* 0;

*return* 1;

}

*void* AC\_suggestionsRec(*struct TrieNode\_AC\* root*,*string currPrefix*)

{

*if* (*root*->isWordEnd)

    {

        AC\_ans.push\_back(*currPrefix*);

    }

*if* (AC\_isLastNode(*root*))

*return*;

*for* (*int* i = 0; i < 26; i++)

    {

*if* (*root*->children[i])

        {

*currPrefix*.push\_back(97 *+* i);

            AC\_suggestionsRec(*root*->children[i], *currPrefix*);

*currPrefix*.pop\_back();

        }

    }

}

*int* AC\_printAutoSuggestions(*TrieNode\_AC\* root*,*const string query*)

{

*struct* TrieNode\_AC*\** pCrawl = *root*;

*int* level;

*int* n = *query*.length();

*for* (level = 0; level < n; level++)

    {

*int* index = *query[*level*]-*'*a*';

*if* (!pCrawl->children[index])

*return* 0;

        pCrawl = pCrawl->children[index];

    }

*bool* isWord = (pCrawl->isWordEnd == true);

*bool* isLast = AC\_isLastNode(pCrawl);

*if* (isWord && isLast)

    {

        AC\_ans.push\_back(*query*);

*return* *-*1;

    }

*if* (!isLast)

    {

        string prefix = *query*;

        AC\_suggestionsRec(pCrawl, prefix);

*return* 1;

    }

}

*int* auto\_comp()

{

*struct* TrieNode\_AC*\** root = getNode();

    ifstream myFile;

    myFile.open("*test.csv*");

*int* row\_count =0 ;

    string line;

*while*(getline(myFile,line))

    {

        row\_count +=1;

        stringstream  lineStream(line);

        string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

        {

            column\_count+=1;

*if*(column\_count==2)

            {

                AC\_insert(root,cell);

            }

        }

    }

    string key;

    cout*<<*"*\n Insert the initials of the contact name :*"*<<*endl;

    cin*>>*key;

    cout*<<*endl*<<*endl*<<*endl;

*int* comp = AC\_printAutoSuggestions(root, key);

*if* (comp == *-*1)

        cout *<<* "*No other Name found with this prefix in the contact list*"*<<*endl;

*else* *if* (comp == 0)

        cout *<<* "*No Name found with this prefix in the contact list\n*";

    cout*<<*"*ID\t\t\tNAME\t\t\tNUMBER\t\t\tOPERATOR\t\tLOCATION*"*<<*endl*<<*endl;

*int* q=AC\_ans.size();

*int* index[q]={*-*1};

*int* p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*int* vai=p;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

                string abc=AC\_ans.at(p);

*if* ( column\_count == 2)

                {

*if*(cell.compare(abc)==0)

                    {

                        index[p]=row\_count;

                        p++;

*break*;

                    }

                }

            }

*if*(vai<p)

            {

*break*;

            }

        }

    }

    q=AC\_ans.size();

    p=0;

*while*(q*--*)

    {

        ifstream myFile;

        myFile.open("*test.csv*");

*int* row\_count =0 ;

        string line;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*((index[p]==row\_count) && (column\_count==1 || column\_count==2 || column\_count==3 || column\_count==4 || column\_count==5))

                {

*if*(cell.size()<8)

                        cout*<<*cell*<<*"*\t\t\t*";

*else*

                        cout*<<*cell*<<*"*\t\t*";

                }

            }

*if*(index[p]==row\_count)

            {

                p++;

                cout*<<*endl;

            }

        }

    }

*return* 0;

}

*struct* TrieNode\_SC {

    TrieNode\_SC*\** Trie[256];

*bool* isEnd;

    TrieNode\_SC()

    {

*for* (*int* i = 0; i < 256; i++) {

            Trie[i] = NULL;

        }

        isEnd = false;

    }

};

*void* SC\_InsertTrie(*TrieNode\_SC\* root*,*string s*)

{

    TrieNode\_SC*\** temp = *root*;

*for* (*int* i = 0; i < *s*.length(); i++) {

*if* (temp->Trie[*s[*i*]*] == NULL) {

            temp->Trie[*s[*i*]*] = *new* TrieNode\_SC();

        }

        temp = temp->Trie[*s[*i*]*];

    }

    temp->isEnd = true;

}

*void* SC\_printSuggestions(*TrieNode\_SC\* root*,*string res*)

{

*if* (*root*->isEnd == true)

    {

        SC\_count++;

        cout*<<*SC\_count*<<*"*:*";

        cout *<<* *res* *<<* endl;

    }

*for* (*int* i = 0; i < 256; i++)

    {

*if* (*root*->Trie[i] != NULL)

        {

*res*.push\_back(i);

            SC\_printSuggestions(*root*->Trie[i], *res*);

*res*.pop\_back();

        }

    }

}

*bool* SC\_checkPresent(*TrieNode\_SC\* root*,*string key*)

{

*for* (*int* i = 0; i < *key*.length(); i++)

    {

*if* (*root*->Trie[*key[*i*]*] == NULL)

        {

            cout*<<*"*NO , The record with same spelling is not present in contact list\n*"*<<*endl*<<*endl*<<*endl;

            cout*<<*"*\nRecords with similar Names :\n*"*<<*endl*<<*endl;

            SC\_printSuggestions(*root*, *key*.substr(0, i));

*return* false;

        }

*root* = *root*->Trie[*key[*i*]*];

    }

*if* (*root*->isEnd == true)

    {

*return* true;

    }

    cout*<<*"*NO , The record with same spelling is not present in contact list*"*<<*endl*<<*endl*<<*endl;

    cout*<<*"*Records with similar Names :*"*<<*endl*<<*endl;

    SC\_printSuggestions(*root*, *key*);

*return* false;

}

*int* spell\_check()

{

    ifstream myFile;

    myFile.open("*test.csv*");

*int* row\_count =0 ;

    string line;

    vector<string> str ;

*while*(getline(myFile,line))

    {

        row\_count +=1;

        stringstream  lineStream(line);

        string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

        {

            column\_count+=1;

*if* ( column\_count == 2)

            {

                str.push\_back(cell);

            }

        }

    }

    myFile.close();

    string key;

    cout*<<*"*\n ENTER THE CONTACT NAME :*"*<<*endl;

    cin*>>*key;

    cout*<<*endl;

    TrieNode\_SC*\** root = *new* TrieNode\_SC();

*for* (*int* i = 0; i < str.size(); i++)

    {

        SC\_InsertTrie(root, str*[*i*]*);

    }

*if* (SC\_checkPresent(root, key))

    {

        cout *<<* "*YES , The record with same spelling is present in contact list\n*"*<<*endl;

        myFile.open("*test.csv*");

*int* row\_count =0,pos=0;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if* ( column\_count == 2)

                {

*if*(cell.compare(key)==0)

                    {

                        pos=row\_count;

*break*;

                    }

                }

            }

        }

        myFile.close();

        myFile.open("*test.csv*");

        row\_count =0;

*while*(getline(myFile,line))

        {

            row\_count +=1;

            stringstream  lineStream(line);

            string cell;

*int* column\_count = 0 ;

*while*(getline(lineStream,cell,'*,*'))

            {

                column\_count+=1;

*if*(row\_count==pos && (column\_count==1 || column\_count==2 || column\_count==3 || column\_count==4 || column\_count==5))

                {

*if*(column\_count==1)

                        cout*<<*"*Id \t\t:*";

*if*(column\_count==2)

                        cout*<<*"*Name \t\t:*";

*if*(column\_count==3)

                        cout*<<*"*Contact Number  :*";

*if*(column\_count==4)

                        cout*<<*"*Operator \t:*";

*if*(column\_count==5)

                        cout*<<*"*Location \t:*";

                    cout*<<*cell*<<*endl;

                }

            }

*if*(row\_count>pos)

*break*;

        }

        myFile.close();

    }

*return* 0;

}

*int* insert\_on\_dataset()

{

    ofstream my;

    my.open("*test.csv*",ios::app);

    total++;

    stringstream ss;

    ss *<<* total;

    string str = ss.str();

    my*<<*str*<<*"*,*";

*int* i=0;

    string a1,a2,a3,a4;

    cout*<<*"*Name \t\t:*"*<<*endl;

    cin*>>*a1;

    cout*<<*"*Contact Number  :*"*<<*endl;

    cin*>>*a2;

    cout*<<*"*Operator \t:*"*<<*endl;

    cin*>>*a3;

    cout*<<*"*Location \t:*"*<<*endl;

    cin*>>*a4;

    my*<<*a1*<<*"*,*"*<<*a2*<<*"*,*"*<<*a3*<<*"*,*"*<<*a4*<<*endl;

    cout*<<*"*\n NEW RECORD INSERTED*"*<<*endl;

*return* 0;

}

*int* delete\_in\_dataset()

{

    fstream fin, fout;

    fin.open("*test.csv*", ios::in);

    fout.open("*test12.csv*", ios::out);

*int* row\_count =0 ;

    string line;

*int* id;

    cout *<<* "*Enter the id of the record to be deleted:*"*<<*endl;

    cin *>>* id;

*int* count=0;

*while*(getline(fin,line))

    {

        row\_count +=1;

        stringstream  lineStream(line);

        string cell;

*int* column\_count = 0 ;

        string a1,a2,a3,a4,a5;

*while*(getline(lineStream,cell,'*,*'))

        {

            column\_count+=1;

*if*(row\_count==id)

            {

                count=1;

                a1*=*"*zzzzz*";

                a2*=*"*zzzzz*";

                a3*=*"*zzzzz*";

                a4*=*"*zzzzz*";

                a5*=*"*zzzzz*";

            }

*else*

            {

*if*(column\_count==1 )

                    a1*=*cell;

*if*(column\_count==2 )

                    a2*=*cell;

*if*(column\_count==3 )

                    a3*=*cell;

*if*(column\_count==4 )

                    a4*=*cell;

*if*(column\_count==5 )

                    a5*=*cell;

            }

        }

        fout*<<*a1*<<*"*,*"*<<*a2*<<*"*,*"*<<*a3*<<*"*,*"*<<*a4*<<*"*,*"*<<*a5*<<*endl;

    }

*if* (count == 1)

        cout *<<* "*Record deleted\n*";

*else*

        cout *<<* "*Record not found\n*";

    fin.close();

    fout.close();

    fstream my;

    my.open("*test.csv*",ios::out *|* ios::trunc);

    my.close();

    fin.open("*test12.csv*", ios::in);

    fout.open("*test.csv*", ios::out);

    row\_count =0 ;

*while*(getline(fin,line))

    {

        row\_count +=1;

        stringstream  lineStream(line);

        string cell;

*int* column\_count = 0 ;

        string a1,a2,a3,a4,a5;

*while*(getline(lineStream,cell,'*,*'))

        {

            column\_count+=1;

*if*(column\_count==1 )

                a1*=*cell;

*if*(column\_count==2 )

                a2*=*cell;

*if*(column\_count==3 )

                a3*=*cell;

*if*(column\_count==4 )

                a4*=*cell;

*if*(column\_count==5 )

                a5*=*cell;

        }

        fout*<<*a1*<<*"*,*"*<<*a2*<<*"*,*"*<<*a3*<<*"*,*"*<<*a4*<<*"*,*"*<<*a5*<<*endl;

    }

    fin.close();

    fout.close();

    my.open("*test12.csv*",ios::out *|* ios::trunc);

    my.close();

*return* 0;

}

*int* operation\_on\_dataset()

{

*int* choice;

    cout*<<*"*\n\n\n\t\t\t\t\t\t  DATASET OPERATIONS*"*<<*endl;

    cout*<<*endl*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 1: Insert  a new entry in dataset*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 2: delete a entry from dataset*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 3: EXIT*"*<<*endl*<<*endl;

    cout*<<*"*------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------*"*<<*endl;

    cin*>>*choice;

*switch*(choice)

    {

*case* 1:

        {

            system("*cls*");

*char* ch='*Y*';

*while*(ch=='*Y*')

            {

                system("*cls*");

                insert\_on\_dataset();

                cout*<<*"*\n\n\n\n\t DO YOU WANT TO INSERT MORE ENTRY ? (Y/N) :*"*<<*endl;

                cin*>>*ch;

            }

*break*;

        }

*case* 2:

        {

            system("*cls*");

            delete\_in\_dataset();

*break*;

        }

*case* 3:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\t\t\t\t\t\tThank you*";

*break*;

        }

*default*:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\n \t\t\t\t\t\tWrong choice entered*"*<<*endl;

*break*;

        }

    }

*return* 0;

}

*int* contact\_list()

{

*int* choice;

    cout*<<*"*\n\n\n\t\t\t\t\t\t   CONTACT LIST OPERATIONS*"*<<*endl;

    cout*<<*endl*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 1: FULL TEXT SEARCH*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 2: SORT THE CONTACT LIST AS PER REQUIREMENT*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 3: AUTO-COMPLETE OPERATION*"*<<*endl*<<*endl;

     cout*<<*"*\n\t\t\t\t\t\t 4: SPELL- CHECKER OPERATION*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 5: EXIT*"*<<*endl*<<*endl;

    cout*<<*"*------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------*"*<<*endl;    cin*>>*choice;

*switch*(choice)

    {

*case* 1:

        {

            system("*cls*");

            Full\_text\_search();

*break*;

        }

*case* 2:

        {

            system("*cls*");

            sort\_op();

*break*;

        }

*case* 3:

        {

            system("*cls*");

            auto\_comp();

*break*;

        }

*case* 4:

        {

            system("*cls*");

            spell\_check();

*break*;

        }

*case* 5:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\t\t\t\t\t\tThank you*";

*break*;

        }

*default*:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\n \t\t\t\t\t\tWrong choice entered*"*<<*endl;

*break*;

        }

    }

*return* 0;

}

*int* main()

{

    ios::sync\_with\_stdio(0);

    cin.tie(0);

    system("*cls*");

*int* choice;

    cout*<<*"*\n\n\n\t\t\t\t\t\t   WELCOME TO Public Call Operations*"*<<*endl;

    cout*<<*endl*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 1: Operations on Dataset*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 2: Access the Contact List*"*<<*endl*<<*endl;

    cout*<<*"*\n\t\t\t\t\t\t 3: EXIT*"*<<*endl*<<*endl;

    cout*<<*"*------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------*"*<<*endl;

    cin*>>*choice;

*switch*(choice)

    {

*case* 1:

        {

            system("*cls*");

            operation\_on\_dataset();

*break*;

        }

*case* 2:

        {

            system("*cls*");

            contact\_list();

*break*;

        }

*case* 3:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\t\t\t\t\t\tThank you*";

*break*;

        }

*default*:

        {

            system("*cls*");

            cout*<<*"*\n\n\n*";

            cout*<<*"*\n \t\t\t\t\t\tWrong choice entered*"*<<*endl;

*break*;

        }

    }

*return* 0;

}

OUTPUT:

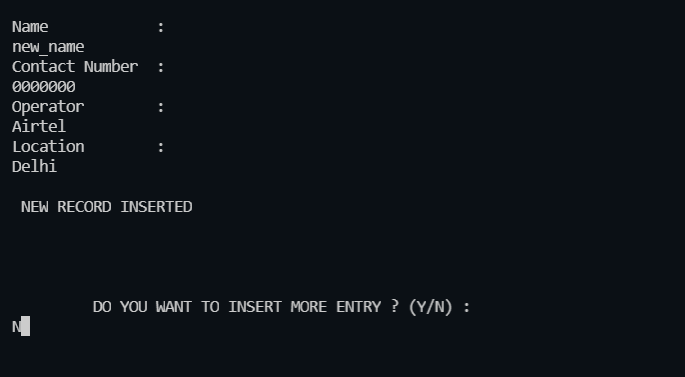
* Main screen



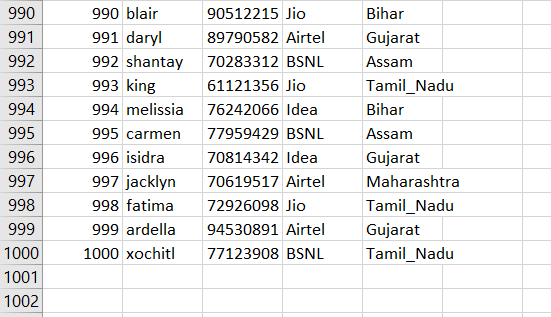
* Operations on dataset



* Insert



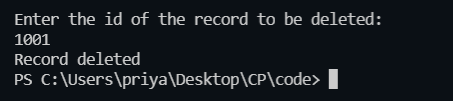
(dataset before insertion)



(dataset after insertion)



* Deletion



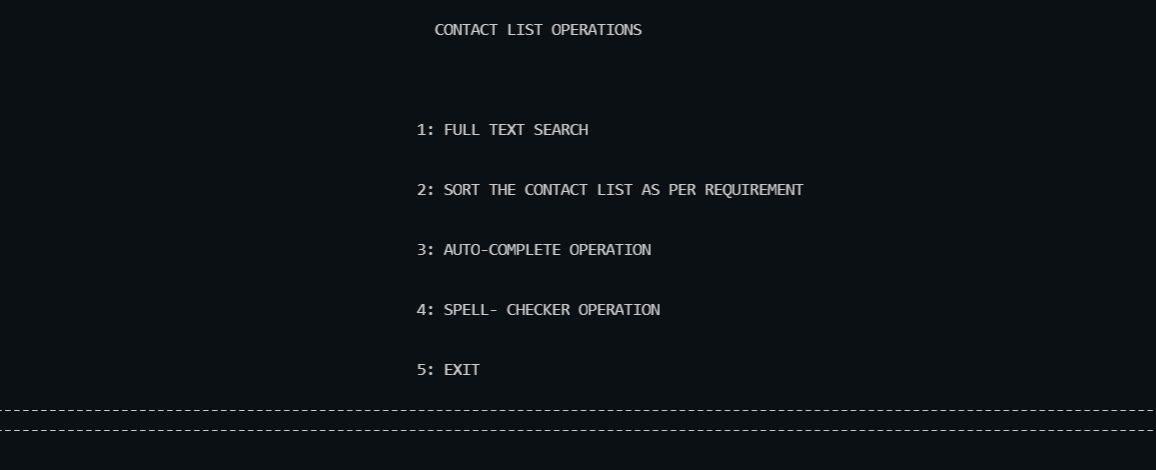
Before deletion



After deletion



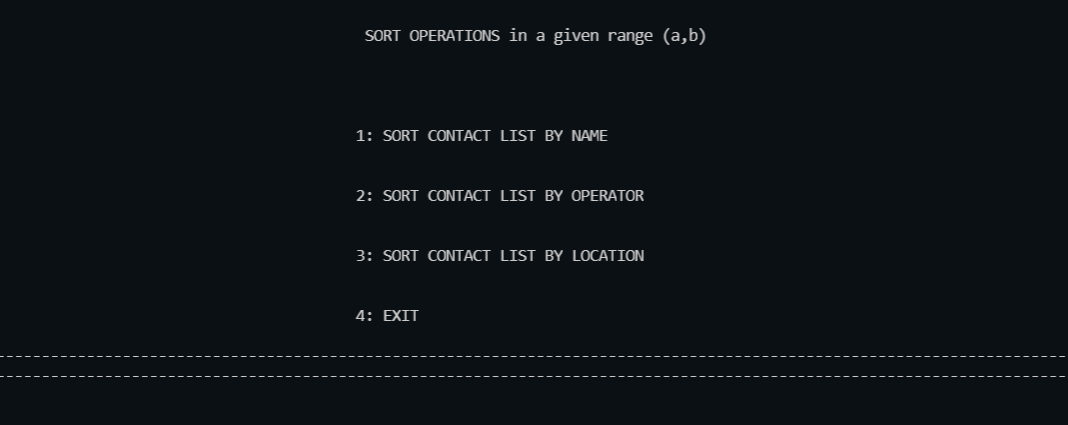
* contact list operations



* full text search



* + sorting



* + - sorting by name



* + - sorting by operator



* + - sorting by location

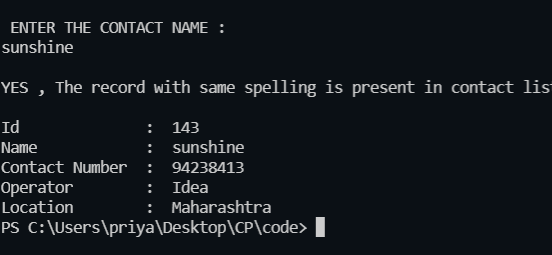


* autocomplete

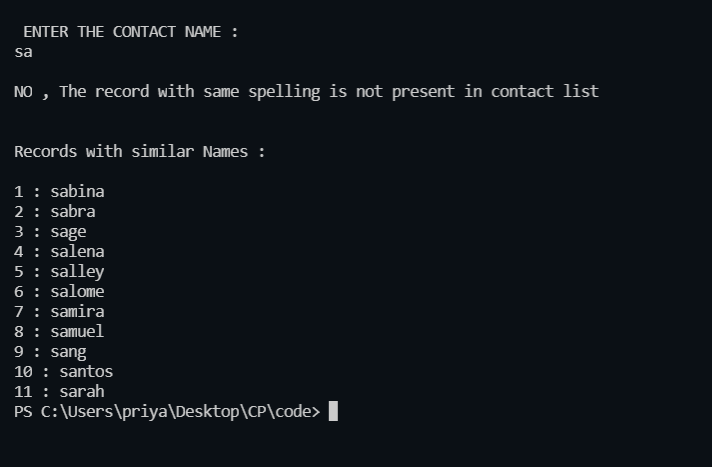


* spell checker

(yes)



(no)



* exit

