C++ FILE
CONSOLEAPPLICATION1.CPP

```
#include <iostream>
#include "Python.h"
#include<fstream>
#include<string>
#include<map>
#include<vector>
#include<algorithm>
using namespace std;



bool sortcol(const vector<long long int>& v1, const vector<long long int>& v2) {
	return v1[0] > v2[0];
}

long long int getPolinomialRollHash(
	string const& str)
{
	int p = 31;
	int m = 1e9 + 9;
	long long int power_of_p = 1;
	long long int hash_val = 0;

	for (int i = 0; i < str.length(); i++) {
		hash_val = (hash_val + (str[i] - 'a' + 1) * power_of_p) % m;
		power_of_p = (power_of_p * p) % m;
	}

	return hash_val;
}


vector<string> getsuggestion(string &movie,string &final) {


	ofstream Mefile("myfile.txt");
	Mefile << movie;
	Mefile.close();

	Py_Initialize();


	PyRun_SimpleString("exec(open('read.py').read())");

	string myText;


	ifstream MyReadFile("myfile1.txt");


	while (getline(MyReadFile, myText)) {

		final += myText;
```

```
            final += " ";
        }


        MyReadFile.close();


        ofstream ffile("finalfile.txt");
        ffile << final;
        ffile.close();

        PyRun_SimpleString("exec(open('cosinepython.py').read())");


        string output;
        vector<string> result;

        ifstream MyoutputFile("output.txt");


        cout << endl;
        cout << "Suggestion based on previous searches : " << endl << endl;
        while (getline(MyoutputFile, output)) {


            result.push_back(output);
        }


        MyoutputFile.close();

        return result;
}


int main()
{
        vector<vector<long long int>> arr(12037, vector<long long int>(3));
        vector<string> arrstr(12037);
        for (int j = 0; j < 12037; j++) {
            arrstr.push_back("*");
        }
        vector<vector<long long int>> arrsort(12037, vector<long long int>(3));

        for (int p = 0; p < 12037; p++)
            for (int f = 0; f < 3; f++)
                arrsort[p][f] = arr[p][f];

        map <string, int> check;
        ifstream Myhastable("Hash.txt");

        int counter = 0;
        string word;
        int r = 0;
        while (!Myhastable.eof()) {


            if (Myhastable >> word) {
```

```
                if (r == 3) {
                        r = 0;
                        counter++;
                }
                arr[counter][r] = stoi(word);
                r++;

        }


}
Myhastable.close();

ifstream Mymovies("movieshash.txt");

int counter1 = 0;
string word1;

while (getline(Mymovies,word1)) {

                arrstr[counter1] = word1;

                counter1++;

}

Mymovies.close();



string final = "";
int i = 0;
while (1) {
        if (i == 0) {
                string movie;
                int rating;
                vector<string> out;

                cout << "Enter a movie you like to watch : ";
                getline(cin, movie);

                cout << "Enter rating : ";
                cin >> rating;
                out = getsuggestion(movie, final);



                string correctmovie;
                string cmovie;
                ifstream Mytitle("title.txt");
                while (getline(Mytitle, cmovie))
                {
                        correctmovie = cmovie;
                }
                check[correctmovie] = 1;
                long long int key = abs( getPolinomialRollHash(correctmovie));

                long long int uniquekey = key;
```

```
                key %= 12037;

                        int counter = 0;
                        while (arr[key][0] != 0 && counter < 12037 && arr[key][1] !=
uniquekey)
                        {
                                key = (key + counter + 12037) % 12037;
                                counter++;
                        }


                arr[key][0] = arr[key][0] + rating;
                arr[key][1] = uniquekey;
                arr[key][2] = key;
                arrstr[key] = correctmovie;

                ofstream Myhash("Hash.txt");
                for (int h = 0; h < 12037; h++) {
                        Myhash << arr[h][0] << " "<<arr[h][1]<<" "<<arr[h][2] << endl;
                }

                for (int p = 0; p < 12037; p++)
                        for (int f = 0; f < 3; f++)
                                arrsort[p][f] = arr[p][f];


                Myhash.close();
                ofstream Mymoviehash("movieshash.txt");
                for (int h = 0; h < 12037; h++) {
                        Mymoviehash << arrstr[h] <<endl;
                }

                Mymoviehash.close();

                for (auto& it : out)
                        cout << it << endl;
        }
        cout << endl;
        cout << "If want to search another movie and improve recommendation list
(TYPE 'y')";
        cout << endl;
        cout << "If want to refresh the recommendation list (TYPE 'r')";
        cout << endl;
        cout << "Exit (TYPE 'x')";
        cout << endl;
        cout << "Trending list (TYPE 't')";
        cout << endl;
        cout << "Enter choice : ";
        string in;

        cin >> in;

        string movie;
        int rating;
        vector<string> out;
```

```
		if (in == "y") {
			cout << endl;

			cin.ignore();
			cout << "Enter a movie you like to watch : ";
			getline(cin, movie);

			cout << "Enter rating : ";
			cin >> rating;

			out = getsuggestion(movie, final);

			string correctmovie;
			string cmovie;
			ifstream Mytitle("title.txt");
			while (getline(Mytitle, cmovie))
			{
				correctmovie = cmovie;
			}
			if (check[correctmovie] == 1) {
				cout << "MOVIE ALREADY RATED IN THIS SESSION" << endl;

			}
			else {
				check[correctmovie] = 1;
				long long int key = abs(getPolinomialRollHash(correctmovie));
				long long int uniquekey = key;

				key %= 12037;

				int counter = 0;
				while (arr[key][0] != 0 && counter < 12037 && arr[key][1] !=
uniquekey)
				{
					key = (key + counter + 12037) % 12037;
					counter++;
				}


				arr[key][0] = arr[key][0] + rating;
				arr[key][1] = uniquekey;
				arr[key][2] = key;
				arrstr[key] = correctmovie;

				ofstream Myhash("Hash.txt");
				for (int h = 0; h < 12037; h++) {
					Myhash << arr[h][0] << " " << arr[h][1] <<"
"<<arr[h][2]<< endl;
				}

				Myhash.close();

				for (int p = 0; p < 12037; p++)
					for (int f = 0; f < 3; f++)
						arrsort[p][f] = arr[p][f];


				ofstream Mymoviehash("movieshash.txt");
```

```
                        for (int h = 0; h < 12037; h++) {
                                Mymoviehash << arrstr[h] << endl;
                        }

                        Mymoviehash.close();



                        for (auto& it : out)
                                cout << it << endl;
                }
        }
        else if (in == "r") {
                final = "";
                cout << "REFRESHED" << endl;
        }
        else if (in == "t") {
                cout << endl;
                cout << "TRENDING LIST : " << endl;
                sort(arrsort.begin(), arrsort.end(), sortcol);
                int k = 0;
                while(arrsort[k][0]){
                        cout << arrstr[arrsort[k][2]] <<" "<<"      |||      Total
Score(Till Date) : "<< arrsort[k][0]<< endl;
                        if (k == 10)
                                break;
                        k++;

                }

        }

        else if (in == "x") {
                exit(0);
        }

        else
                cout << "INVALID INPUT" << endl;
        i++;
    }

}
```

PYTHON FILES

READ.PY

```
import pandas as pd
from fuzzywuzzy import fuzz
from fuzzywuzzy import process
file1 = open("myfile.txt","r")
```

```
x=file1.read()
file1.close()
df=pd.read_csv("main1_data.csv")

highest = process.extractOne(x,df["movie_title"].to_list())



df['index'] = df.index

def get_title_from_index(index):
    return df[df.index==index]["movie_title"].values[0]

def get_comb_from_index(index):
    return df[df.index==index]["comb"].values[0]
def get_index_from_title(title):
    return df[df.movie_title==title]["index"].values[0]

f=get_index_from_title(highest[0])

print("DO YOU MEAN : ")
print(highest[0])
file3=open("title.txt","w")
file3.write(highest[0])

y=get_comb_from_index(f)
file3.close()
file2 = open("myfile1.txt","w")
file2.write(y)
file2.close()
```

COSINEPYTHON.PY

```
import pandas as pd
df=pd.read_csv("main1_data.csv")
file1 = open("finalfile.txt","r")
x=file1.read()
file1.close()
movie_user_likes=x
```

```
df2 = {'director_name': '', 'actor_1_name': '', 'actor_2_name': '','actor_3_name'
:'','genres':'','movie_title':'','comb':movie_user_likes}
df = df.append(df2,ignore_index = True)
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
cv=CountVectorizer()
count_matrix=cv.fit_transform(df["comb"])

cosine_sim=cosine_similarity(count_matrix)
Similar_movies= list(enumerate(cosine_sim[6010]))
sorted_similar_movies= sorted(Similar_movies,key= lambda x:x[1],reverse= True)[1:
]
df['index'] = df.index

def get_title_from_index(index):
    return df[df.index==index]["movie_title"].values[0]
def get_index_from_title(title):
    return df[df.movie_title==title]["index"].values[0]
file2 = open("output.txt","w")

i=0;
for element in sorted_similar_movies:
    file2.write(get_title_from_index(element[0]))
    file2.write('\n')
    i=i+1;
    if i>=21:
        break

file2.close()
```

RESULTS

UPDATING SUGGESTIONS BASED ON THE MOVIES USER ENTERED

```
Enter a movie you like to watch : specter
Enter rating : 3
DO YOU MEAN :
spectre

Suggestion based on previous searches :


skyfall
quantum of solace
the legend of tarzan
alita: battle angel
stealth
iboy
the green hornet
xxx
cliffhanger
executive decision
shadow conspiracy
for your eyes only
spider-man 3
the wild bunch
the hunter's prayer
adrift
pirates of the caribbean: on stranger tides
spider-man 2
mission: impossible - rogue nation
mission: impossible iii

If want to search another movie and improve recommendation list (TYPE 'y')
If want to refresh the recommendation list (TYPE 'r')
Exit (TYPE 'x')
Trending list (TYPE 't')
Enter choice : y
```

```
Enter a movie you like to watch : pirates
Enter rating : 4
DO YOU MEAN :
pirates of the caribbean: at world's end

Suggestion based on previous searches :

pirates of the caribbean: at world's end
spectre
pirates of the caribbean: dead man's chest
pirates of the caribbean: the curse of the black pearl
skyfall
the lone ranger
pirates of the caribbean: on stranger tides
the lord of the rings: the fellowship of the ring
the lord of the rings: the return of the king
the lord of the rings: the two towers
quantum of solace
pirates of the caribbean: dead men tell no tales
spider-man 2
spider-man
the three musketeers
dr. no
snow white and the huntsman
the legend of tarzan
the mummy returns
the scorpion king
in the name of the king: a dungeon siege tale
```

TRENDING LIST BASED ON RATINGS GIVEN BY ALL THE USERS TILL DATE

```
If want to search another movie and improve recommendation list (TYPE 'y')
If want to refresh the recommendation list (TYPE 'r')
Exit (TYPE 'x')
Trending list (TYPE 't')
Enter choice : t

TRENDING LIST :
spectre       |||      Total Score(Till Date) : 22
the matrix       |||      Total Score(Till Date) : 18
the space between us       |||      Total Score(Till Date) : 11
frozen       |||      Total Score(Till Date) : 8
shade       |||      Total Score(Till Date) : 8
tangled       |||      Total Score(Till Date) : 8
deadpool       |||      Total Score(Till Date) : 7
avatar       |||      Total Score(Till Date) : 5
the wolf of wall street       |||      Total Score(Till Date) : 5
last christmas       |||      Total Score(Till Date) : 5
red       |||      Total Score(Till Date) : 4
```

REFRESHING\DELETING THE HISTORY OF THE USER

```
If want to search another movie and improve recommendation list (TYPE 'y')
If want to refresh the recommendation list (TYPE 'r')
Exit (TYPE 'x')
Trending list (TYPE 't')
Enter choice : r
REFRESHED
```