Indian Institute of Technology, Guwahati

Department of Computer Science and Engineering

Project Report

On

“Speech Based Browser Automation”

Based on

Speech Recognition System

Course: CS566 Speech Processing

GROUP NO 19

Submitted to

Prof.

P.K..Das

Submitted by:

Keshav Parihar (214101025)

Mohit Kumar (214101029)

TABLE OF CONTENT

ABSTRACT

This document defines a set of evaluation criteria and test methods for speech recognition systems used in searching and retrieving contact details. This report is on the project which detects the contact name and show its details

.

INTRODUCTION

In this report, we concentrate on the speech recognition programs that are human-computer interactive. When software evaluators observe humans testing such software programs, they gain valuable insights into technological problems and barriers that they may never witness otherwise. . Testing speech recognition products for universal usability is an important step before considering the product to be a viable solution for its customers later. This document concerns Speech Recognition accuracy in contact searching and retrieving details, which is a critical factor in the development of hands-free human- machine interactive devices. There are two separate issues that we want to test: word recognition accuracy and software friendliness. Major factors that impede recognition accuracy in the environment noise sources and system noise.

However, what is speech recognition?

Speech recognition works like this. You speak into a microphone and the computer transforms the sound of your words into text to be used by your word processor or other applications available on your computer. The computer may repeat what you just said or it may give you a prompt for what you are expected to say next. This is the central promise of interactive speech recognition. You also had to correct any errors virtually as soon as they happened, which means that you had to concentrate so hard on the software that you often forgot what you were trying to say.

The new voice recognition systems are certainly much easier to use. You can speak at a normal pace without leaving distinct pauses between words. However, you cannot really use "*natural speech*" as claimed by the manufacturers. You must speak clearly, as you do when you speak to a Dictaphone or when you leave someone a telephone message. Remember, the computer is relying solely on your spoken words. It cannot interpret your tone or inflection, and it cannot interpret your gestures and facial expressions, which are part of everyday human communication. Some of the systems also look at whole phrases, not just the individual words you speak. They try to get information from the context of your speech, to help work out the correct interpretation.

The goal of this project is to define a set of evaluation criteria and test methods for the interactive voice recognition systems for searching contact and retrieving corresponding details for successful search.

PROPOSED METHODOLOGY

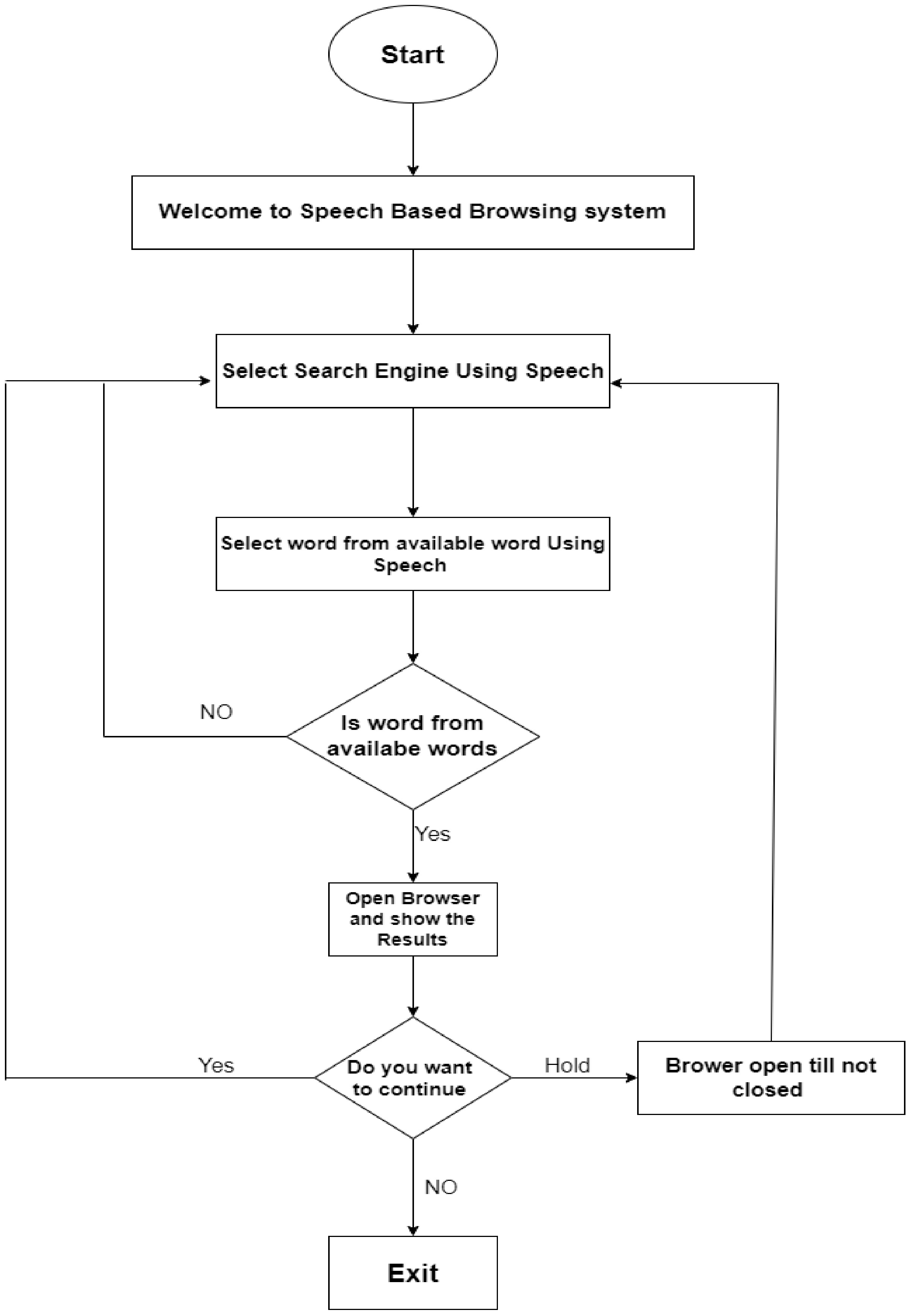
Basic requirements to develop this project are as follows:

- ✓ Windows OS
- ✓ Microsoft Visual Studio 2010
- ✓ C++ 11 integrated with VS2010
- ✓ Recording Module

With the availability of above software, we further proceed in modelling the logic. The prerequisites of this project are

- ✓ Basic i/o operations on file
- ✓ Pre-processing of speech data
- ✓ Feature extraction
- ✓ Modelling of extracted feature
- ✓ Enhancing model

With the availability of above tools, we further proceeded. Below is the flow chart for our project


Start
Welcome to Speech Based Browsing system
Select Search Engine Using Speech
Select word from available word Using Speech
NO
Is word from availabe words
Yes
Open Browser and show the Results
Yes
Do you want to continue
Hold
Brower open till not closed
NO
Exit

EXPERIMENTAL SETUP

This project is divided into following modules:

1. **Training Module**
2. **Testing Module**

1. Training Module

The flow for training over data is as follows:

i. Record the data as 30 utterance of each word

ii. Extract frames for every utterance

iii. Using local distance analysis (in vector quantization) calculate the observation sequence.

iv. Pass this observation sequence to HMM for model designing.

v. Now enhance the model using HMM re-estimation algorithm.

Now reference model is ready for our project. The training of data is not integrated with GUI application. This is different module, which will just evaluate reference model.

2. Testing Module

System will give instruction what is going on and user is required to follow it. The flow of testing is as follows:

i. Live recording of data is done when system instruct.

ii. Testing the data with retrained models.

iii. Detect the Search Engine

iv. Detect the word which has to be search

v. If word is correctly spoken then it opens browser.

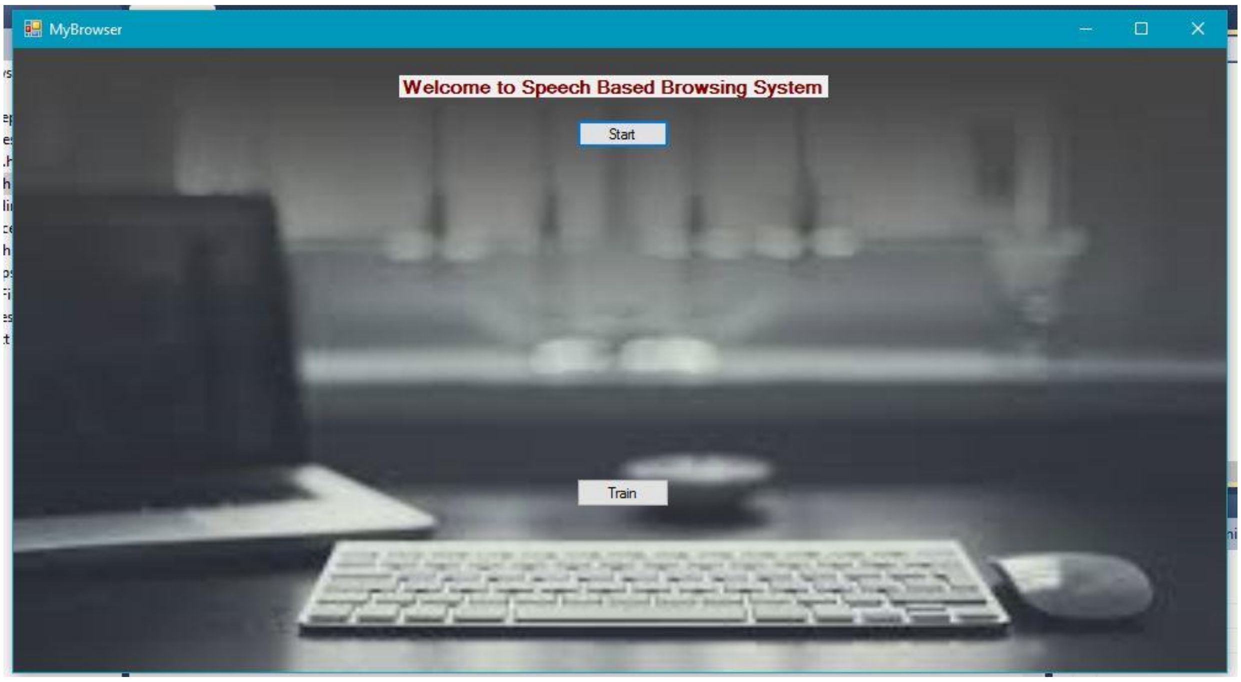
vi. If wrong word detected then ,record the input again.

3. WORDS USED

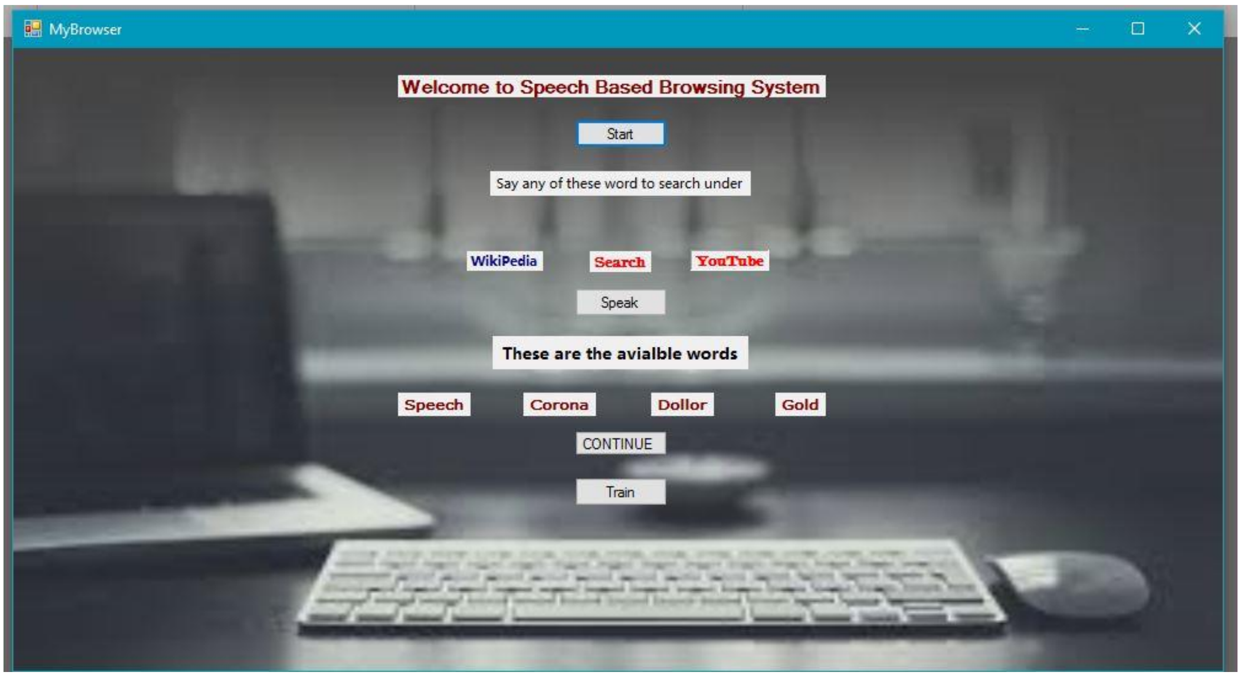
1. **START**
2. **STOP**
3. **YES**
4. **SEARCH**
5. **WIKIPEDIA**
6. **YOUTUBE**
7. **SPEECH**
8. **GOLD**
9. **CORONA**
10. **DOLLAR**

4. SCREEN SHORTS

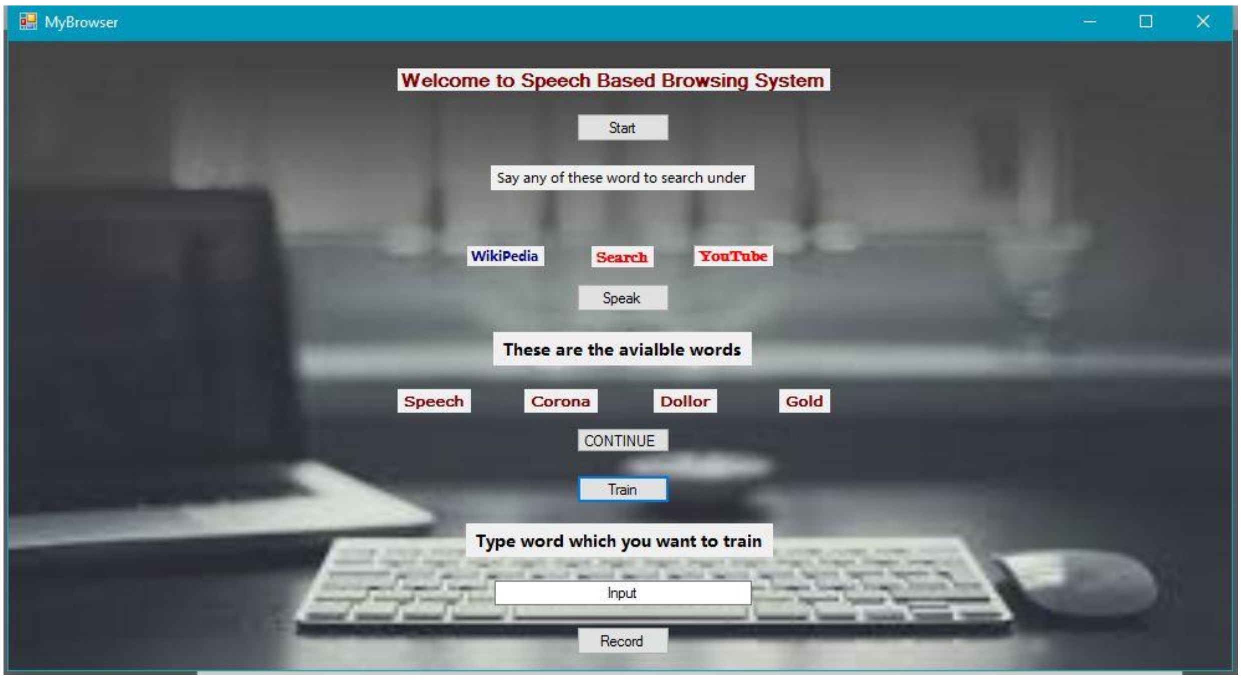
INITIAL WINDOW



WINDOW AFTER START


MyBrowser
Welcome to Speech Based Browsing System
Start
Say any of these word to search under
WikiPedia
Search
YouTube
Speak
These are the avialble words
Speech
Corona
Dollor
Gold
CONTINUE
Train

CLICK TARIN BUTTON


MyBrowser
Welcome to Speech Based Browsing System
Start
Say any of these word to search under
WikiPedia
Search
YouTube
Speak
These are the avialble words
Speech
Corona
Dollor
Gold
CONTINUE
Train
Type word which you want to train
Input
Record

RESULT

For offline testing, we took 30 recordings of each word, with deterministic difference of maximum and second maximum P (O/lambda) we got 85% accuracy and without considering deterministic difference we got 95% accuracy.

The Spoken word will open in browser with selected Search engine i.e. YouTube , Google or Wikipedia , Then we search for the available words.

SOURCE CODE

```
#include "stdafx.h"
#include<iostream>
#include<iomanip>
#include<fstream>
#include<vector>
#include<string>
#include<ctype.h>
#include<string.h>
#include<cstring>

using namespace std;
typedef  long double ld;


////////////////////////////// VARIABLES USED FOR HMM
/////////////////////////////////////////////////////////////////
#define N 5                                                                 // number of states
#define M 32                                                                // number of distinct observation symbols
#define T 40                                                                // number of frames
#define p 12                                                                // number of capstral coefficients per
frame
#define F 320                                                               // size of a frame (320 samples)

vector<vector<long double>> cb;                                             // codebook 32x12
vector<ld> data;
vector<ld> ob;                                                              // observation sequence
vector<ld> pi(N,0);                                                         // initial state distribution
vector<vector<ld>> a( N,vector<ld> (N,0));                                  // state transition probability
distribution
vector<vector<ld>> b( N,vector<ld> (M,0) );                                 // observation symbol probability in state
Sj


vector<ld> pi_sum(N,0);
vector<vector<ld>> a_sum(N,vector<ld> (N,0));
vector<vector<ld>> b_sum(N,vector<ld> (M,0) );


vector<ld> pi_bar(N,0);
vector<vector<ld>> a_bar(N,vector<ld> (N,0));
vector<vector<ld>> b_bar(N,vector<ld> (M,0) );

vector<vector<ld>> alpha(T,vector<ld> (N,0));                              // FORWARD VAIABLE - probab. of partial
onbserv. seq. o1,o2...ot
vector<vector<ld>> beta(T,vector<ld> (N,0));                               // BACKWARD VARIABLE - joint probab. of
ot+1,ot+2....oT at t ans state Si and model λ
vector<vector<ld>> delta(T,vector<ld> (N,0));                              // best score (heighest prob.) along a
single path at time t
vector<vector<ld>> psi(T,vector<ld> (N,0));
vector<vector<ld>> gama(T,vector<ld> (N,0));
vector<vector<vector<ld>>> zeta(T,vector<vector<ld>> (N,vector<ld> (N,0))); // ξ

vector<ld> qstar(T);
ld P,
   pstar;

///////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

////////////////////////////// VARIABLES USED FOR CODEBOOK CALCULATION ///////////////////////////////////////////////

    string  read_line ;//TO READ LINES FROM FILES


         ld
                            sampleval_x1=0,                                                           // FOR
VALUES OF FILE
                            headerline_count=0,                                                       //
READING HEADER FILE FOR NUMBER OF SAMPLE
                            normalised_value=0,
//NORMALISED VALUES
                            // b=0,
                            no_of_frames=0,                                                           // NO OF
FRAMES
                            l=0,
                            i=1,
                            min=0,
//MINIMUM OF WHOLE DATA
                            max_of_data=0,
//MAXIMUM OF WHOLE DATA
                            steadyframes_xi[100][320]={{0}} ,                                                   //STEADY
FRAME MATRIX CONTAINS VALUES OF 5 STEADY FRAMES

                            feature_ref_matrix[41][13]={{0}},                                                   //
FEATURE REFERENCE MATRIX CONTAINS VALUES OF RAISED SINE Ci VALUES

            E=0 ,                                                                                   //USED IN DURBIB
ALGORITHM
                            k[13]={0,0,0,0,0,0,0,0,0,0,0,0,0} ,                                                 //USED
IN DURBIB ALGORITHM
                            sum=0,                                                                              //USED
IN DURBIB ALGORITHM
                            alpha_durbin[13][13]={{0}},                                                         //USED
IN DURBIB ALGORITHM
                            ri[13]={0,0,0,0,0,0,0,0,0,0,0,0,0,0} ,//
                            ai[13]={0,0,0,0,0,0,0,0,0,0,0,0,0} ,//
                            ci[13]={0,0,0,0,0,0,0,0,0,0,0,0,0} ,//
                            raisedci[13]={0,0,0,0,0,0,0,0,0,0,0,0,0} ,
//RAISED SINE Ci VALUES
                            resultant_feat_ref_maxtrix[6][13]={{0}},
//RESULTANT FEATURE REFERENCE MATRIX CONTAINS VALUES OF RAISED SINE Ci VALUES
                            tohkurad=0.0,
//TOKHURA DISTANCE
                            tokhura_w[12]={1.0 ,3.0, 7.0, 13.0 ,19.0, 22.0 , 25.0,33.0,42.0,50.0,56.0,61.0},     //GIVEN
TOKURA WEIGHTS
            dc_shift=0.0;                                                                           //DC SHIFT VALUE



         int     line_count=0 ,                                                                         // USE IN
COUNTING THE INDEX VALUES
                            maxindex=0;                                                                          //INDEX
OF MAXIMUM VALUE IN NORMALISED VALUES FOR SEARCHING THE 5 5 STEADY FRAMES

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    void dcshift(string &filename)
    {
        fstream f;
        f.open(filename ,ios::in);
        if(!f)cout<<"ERROR FOR DC SHIFT ::"<<filename<<"\n";
        else
        {
            // cout<<"DC SHIFT ::"<<filename<<"\n";
            while(line_count<=5*F)  // TAKING FIRST 5 FRAMES
            {
                //cout<<i<<"\n";i++;
                f>>read_line;
                sampleval_x1=stold(read_line);
                dc_shift=dc_shift+sampleval_x1;
                //cout<<i-10<<" )"<<dc_shift<<"\n";
                line_count++;
            }
        }
                   dc_shift=dc_shift/1600.0;
              //  cout<<"DC_SHIFT= "<<dc_shift<<"\n";
                   line_count=0;
                   f.close();

    }

    void findmax(string &filename)
    {
        max_of_data=0;
        line_count=0;
        maxindex=0;
        fstream f;
        f.open(filename ,ios::in);
        if(!f)cout<<"ERROR FOR FIND MAX ::"<<filename<<"\n";
        else
        {
            // cout<<"FIND MAX ::"<<filename<<"\n";
            while(!f.eof())
            {
                f>>read_line;
                line_count++;
                sampleval_x1=stold(read_line);

                if((max_of_data >= sampleval_x1))
                {
                    max_of_data=max_of_data;

                }
                else
                {
                    max_of_data=sampleval_x1;
                    maxindex=line_count;
                }
            }
            line_count=0;
            // cout<<"\nMAX= "<<" "<<max_of_data; // PRINTING MAX  IN GIVEN FILE
            // cout<<"\nMAX INDEX = "<<" "<<maxindex;
        }
        f.close();
    }
    ////////////////////////////////FINDING STEADY FRAMEs /////////////////////////////////////////////////////////////
    void findsteadyframes(string &filename)
    {

        fstream f;
        f.open(filename,ios::in);
        if(!f)cout<<"\nERROR FOR FIND STEADY FRAMES ::"<<filename<<"\n";
        else
        {
            // cout<<"\nFIND STEADY FRAMES ::"<<filename<<"\n";
            while(!f.eof())
            {
                f>>read_line;
                                      sampleval_x1=stold(read_line);
                                      line_count++;
                                      if(line_count == (maxindex- 20*F))// TAKING 30 FRAMES LEFT OF MAX VALUE AND 70
FRAMES RIGHT OF MAX VALUE
                                      {
                                                for (int b = 0; b < 40 ; b++)
                                                {
                                                          for(int d=0 ; d < 320 ; d++)
                                                          {
                                                                    f>>read_line;
                                                                    sampleval_x1=stold(read_line);
                                                                    // cout<<sampleval_x1<<"\n";
                                                                    steadyframes_xi[b][d]=sampleval_x1;// START SAVING
STEADY FRAMES VALUES IN A MATRIX OF 5 x 12
                                                                    // cout<<steadyframes_xi[b][d]<<" ";
                                                          }
                                                          // cout<<"---------------------------------\n";
                                                }
                                                break;


                                      }

                //  if(sampleval_x1 >= (50/100)*max_of_data || sampleval_x1 <= -(50/100)*max_of_data )
                //  {
                //         f1<<read_line<<"\n";

                //  }

            }//while ends......
        }
        f.close();
        //NORMALISE////////////////////////////////////////////////
        for (int b = 0; b < 40 ; b++)
                   {
                            for(int d=0 ; d < 320 ; d++)
                            {
                steadyframes_xi[b][d]= ((steadyframes_xi[b][d] - dc_shift ))*(5000/ (max_of_data));

                            }

                   }
        ////////////////////////////////////////////////////////////

        //////////////////////////////////////////////////////////// checking the cut.......
        fstream f1;
        f1.open("log.txt",ios::out);
        if(!f1)cout<<"\nerror...log"<<"\n";
        else
        {
            f1.clear();
            // cout<<"\nsucees...log";
            for (int b = 0; b < 40 ; b++)
                                                {
                                                          for(int d=0 ; d < 320 ; d++)
                                                          {
                                                                    f1<<steadyframes_xi[b][d]<<"\n";
                                                                    // sampleval_x1=stold(read_line);
                                                                    // cout<<sampleval_x1<<"\n";
                                                                    // steadyframes_xi[b][d]=sampleval_x1;// START SAVING
STEADY FRAMES VALUES IN A MATRIX OF 5 x 12
                                                                    // cout<<steadyframes_xi[b][d]<<" ";
                                                          }
```

```
                                                  // cout<<"---------------------------------\n";
                                        }



        }
        ////////////////////////////////////////////////////////////


    }

    void ci_cal()
    {

        for (int b = 0; b < 40 ; b++)
                              {

                                  // cout<<"\nFRAME -->"<<b+1;

                                        ////////////////////////////////// calculating ri
////////////////////////////////////////////////////////
                                        // cout<<"\nCALCULATING R(i).....\n";
                    ld sum=0;
                                        for (int i = 0; i <= 12; i++)
                                        {
                                                  // if(i==1)break;
                                                  sum=0;
                                                  for (int j = 0; j <= 320 -1 -i ; j++)
                                                  {
                                                            sum = sum + steadyframes_xi[b][j] *
steadyframes_xi[b][i+j];

                                                            // cout<<j+1<<" "<<test[j]*test[i+j]<<"\n";
                                                            // cout<<i<<" "<<i+j<<"\n";

                                                  } //cout<<"------------------------------------------\n\n";
                                                  feature_ref_matrix[b][i] = sum;
                                                  // cout<<a<<"\n";

                                        }

                                        // for (int i = 0; i <= 12; i++)
                                        //    {
                                        //              cout<<b<<" "<<i<<"
"<<fixed<<setprecision(4)<<feature_ref_matrix[b][i]<<"\n";
                                        //                   cout<<b<<" "<<i<<"
"<<feature_ref_matrix[b][i]<<"\n";
                                        //    }
                    //    cout<<"\n";

         ////////////////////////////////////////////////////////////////////////////////////////////////////////////

                    // //////////////////// ai
////////////////////////////////////////////////////////////////////////////////////

                                        // cout<<"\nCALCULATING A(i).....\n";
                                        E=feature_ref_matrix[b][0];
                                        // cout<<fixed<<setprecision(8);
                                        // cout<<E;
                    sum=0;
                                        for (int i = 1; i <= 12 ; i++)
                                        {
                                                  sum = 0.0;
                                                  // cout<<"\n\ni="<<i;
                                                  // cout<<"\nki -------------\n";

                                                            for (int j = 1; j <= i - 1; j++)
                                                            {
                                                                      sum = sum + alpha_durbin[i - 1][j] *
feature_ref_matrix[b][i - j ];

                                                                      // cout<<"sum= "<<sum<<"\n";
                                                            }

                                                            // k[i] calculation
                                                            k[i] = (feature_ref_matrix[b][i] - sum)  / E;
                                                            // cout<<"ki ="<<k[i]<<"\n";

                                                            // cout<<"alpha i i-------------\n";




                                                  alpha_durbin[i][i] = k[i];
                                                  // cout<<alpha[i][i]<<"\n";

                                                  // cout<<"alpha i j-------------\n";


                                                  //alpha calculation
```

```
                                                            for (int j = 1; j <= i - 1; j++)
                                                            {
                                                                    alpha_durbin[i][j] = alpha_durbin[i - 1][j] - ( k[i] *
alpha_durbin[i - 1][i - j] );
                                                                    // cout<<alpha_durbin[i][j]<<"\n";
                                                            }

                                                            // cout<<"E -------------\n";

                                                            //Energy calculatation
                                                            E = (1 - k[i] * k[i]) * E;
                                                            // cout<<E<<"\n";

                                                            // cout<<"-------------\n";


                                                    }//ppppp


                                                    for (int i = 1; i <= 12; i++)
                                                    {
                                                            feature_ref_matrix[b][i-1]=alpha_durbin[12][i];
                                                            // cout<<feature_ref_matrix[b][i]<<"\n";
                                                            // cout<<alpha[12][i]<<"\n";

                                                    }
                                                    //    cout<<"------------------------\n";

                                                    //     for (int i = 1; i <= 12; i++)
                                                    //     {
                                                    //                  //  cout<<feature_ref_matrix[b][i]<<"\n";
                                                    //             cout<<b<<" "<<i<<"
"<<fixed<<setprecision(4)<<feature_ref_matrix[b][i]<<"\n";
                                                    //     }

                                                    //
////////////////////////////////////////////////////////////////////////////////////////////////////////////


                                                    // ////////////////// ci
/////////////////////////////////////////////////////////////////////////////////////////
                                                    // cout<<"\nCALCULATING C(i).....\n";
                                                            // long double ci[13] ,ai[13] ;
                                                            // ci[0]=log(feature_ref_matrix[b][0] *
feature_ref_matrix[b][0]);

                                                            ci[0] = feature_ref_matrix[b][1];
                                                            // cout<<ci[0];


                                                            // long double a=0.0;
                        sum=0;
                                                             for (int m = 1; m <= 12; m++)
                                                             {
                                                                 sum=0;
                                                                 for (int  k = 1; k <= m-1  ; k++)
                                                                 {
                                                                    //  cout<<k<<" "<<  (double)k/ (double)m <<" "<<ci[k]<<"
"<<ai[m-k-1]<<"\n";

                                                                     sum= sum + ((double)k/ (double)m )* ci[k] *
feature_ref_matrix[b][m-k-1];

                                                                    // cout<<(k/m) * ci[k] * ai[m-k]<<"\n";
                                                                    //  cout<<a<<"\n";
                                                                 }//cout<<"---------------\n";

                                                                 ci[m] = feature_ref_matrix[b][m-1] + sum;
                                                                //  cout<<ci[m]<<"\n";

                                                             }

                                                             for (int i = 1; i <= 12; i++)
                                                            {
                                                                    feature_ref_matrix[b][i]=ci[i];
                                                                // cout<<ci[i]<<"\n";
                                                                    // cout<<feature_ref_matrix[b][i]<<" ";

                                                            }
                                                            // cout<<"------------------------\n";

         /////////////////////////////////////////////////////////////////////////////////////////////////////

                                                    //       ////////////////// RAISED SINE
////////////////////////////////////////////////////////////////////////
                                                            // cout<<"\nRAISED SINE RC(i).....\n";


                                                             for (int i = 1; i <= 12; i++)
                                                             {
```

```
                                                            raisedci[i] = feature_ref_matrix[b][i] * (1 + 6 * sin( (
3.1415926535 * i )/12 ) ) ;
                                                        }

                                                        for (int i = 1; i <= 12; i++)
                                                        {
                                                                 feature_ref_matrix[b][i]=raisedci[i];
                                                           //  cout<<raisedci[i]<<"\n";
                                                                // cout<<feature_ref_matrix[b][i]<<" ";
                                                        }
                                                       // cout<<"-----------------------\n";
                                              //
         ///////////////////////////////////////////////////////////////////////////////////////////////////////

                    //// adding in myuniverse.txt

                    // fstream u;
                    // u.open("projectuniverse.txt" , ios::app);
                    // if(!u)cout<<"\nERROR OPENING myuniverse.txt\n";
                    // else
                    // {
                    //      for (int i = 1; i <= 12; i++)
                                              //        {
                                              //                 //  feature_ref_matrix[b][i]=raisedci[i];
                                              //           //  cout<<raisedci[i]<<"\n";
                                              //                 u<<feature_ref_matrix[b][i]<<" ";
                                              //        }
                    //      u<<"\n";
                    //     // cout<<"\nsucess...universe";
                    // }

                }//LOOP ENDS FOR ALL FRAMES...........

    }//ci_cal() end.......

    void findobservationseq()
    {
        int min_tok=INT_MAX;
        int ob_index=0;

            // cout<<"sucess...codebook\n";

            // for (int frame = st; frame < st+1; frame++)
            for (int frame = 0; frame < 40; frame++)
            {
                min_tok=INT_MAX;
                // tohkurad=INT_MAX;
                fstream f;
                f.open("projectcodebook2.txt" ,ios::in|ios::out);
                if(!f)cout<<"\nERROR IN OPENING CODEBOOK \n";
                else
                {
                    for (int row = 0; row < 32; row++)
                    {


                        for (int c = 1; c <= 12; c++)
                        {
                            f>>read_line;
                            // cout<<read_line<<" ";
                            // cout<<feature_ref_matrix[frame][c]<<" ";
                            tohkurad = tohkurad + tokhura_w[c-1] * ( (feature_ref_matrix[frame][c] -
stold(read_line))*(feature_ref_matrix[frame][c] - stold(read_line)) );
                        }
                        // cout<<tohkurad<<"\n";

                        if(tohkurad < min_tok)
                        {
                            min_tok=tohkurad;
                            ob_index=row;
                            // cout<<row<<"\n";
                        }
                        tohkurad=0.0;
                    }

                    ob.push_back(ob_index);
                    // cout<<"\n-----------------------------------------------\n";

                }



            }

        // for (int i = 0; i < ob.size(); i++)
        // {
        //     // cout<<i<<"-->"<<ob[i]<<"\n";
        //     cout<<ob[i]<<" ";
        // }
        // cout<<"\n-----------------------------------------\n";

    }//findobservationseq() ends...................





////////// INITIALISING WITH BAYKIS-MODEL ////////////////////////////////////////////////////////////////////////////
    void baykis_model()
    {
        pi[0]=1.0;
        for(int i=0;i<N;i++)
        {
            for(int j=0;j<N;j++)
            {
                if(i==j&&i!=N-1)
                {
                    a[i][j]=0.8;
                }
                else if(i==j&&i==N-1)
                {
                    a[i][j]=1;
                }
                else if(j==i+1)
                {
                    a[i][j]=0.2;
                }
                else
                    a[i][j]=0;
            }

        }

        for(int i=0;i<N;i++)
        {
            for(int j=0;j<M;j++)
               {
                    b[i][j]=1.0/M;
               }
        }
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void display_A_B_matrix()
{
    cout<<"Pi is \n";
    cout<<"--------------------------------------------------------\n\n";
    for (int i = 0; i < N; i++)
    {
        cout<<pi[i]<<" ";
    }cout<<"\n";
    cout<<"--------------------------------------------------------\n\n";

         cout<<"A matrix is \n";
         cout<<"--------------------------------------------------------\n\n";
         for(int i=0;i<N;i++)
                  {for(int j=0;j<N;j++)
                   cout<<setprecision(12)<<a[i][j]<<"\t";
                   cout<<"\n";
                  }
         cout<<"\n--------------------------------------------------------";
         cout<<"\n"<<"\n";
         cout<<"\n\nB matrix is \n";
         cout<<"\n---------------------------------------------------------------------------------------------------
-----------------\n";
         for(int i=0;i<N;i++)
                  {
           for(int j=0;j<M;j++)
                        cout<<setprecision(12)<<b[i][j]<<"\t";
                     cout<<"\n\n";
                  }
         cout<<"\n---------------------------------------------------------------------------------------------------
------------------";
}


//////////////////////// SOLUTIONS TO 3 PROBLEMS ///////////////////////////////////////////////////////////////////////

    //SOLUTION TO 1 PROBLEM
    ld forward()  // O(N^2 T)
    {
        //initialisation
        for(int i=0;i<N;i++)
        {
            // cout<<pi[i]<<"x"<<b[i][ob[0]-1]<<"\n";
            alpha[0][i]=pi[i]*b[i][ob[0]];
        }


        // cout<<"\n aftr intialization\n";
        // for(int i=0;i<N;i++)
        //     cout<<alpha[0][i]<<"  ";
        // cout<<"\n";

        //induction
        for(int t=0;t<T-1;t++)
        {
            // cout<<t<<"--> ";
            for(int j=0;j<N;j++)
            {

                ld sum=0;
                for(int i=0;i<N;i++)
                {
                    sum+=alpha[t][i] * a[i][j];
                }
                // cout<<sum<<" ";
                //cout<<t<<"   "<<j<<"\n";
                alpha[t+1][j]=sum * b[j][ob[t+1]];
                // cout<<alpha[t+1][j]<<"  ";
            }
            // cout<<"\n";
        }

        //termination
        P=0;
        for(int i=0;i<N;i++)
        {
            P+=alpha[T-1][i];
            //cout << alpha[T-1][i] << "\n";
        }
        return P;
    }

    ld backward() // O(N^2 T)
    {
        // initialisation
        for(int i=0;i<N;i++)
            beta[T-1][i]=1;


        // cout<<"\n after intialization\n";
        // for(int i=0;i<N;i++)
        //     cout<<alpha[0][i]<<"  ";
        // cout<<"\n";


        //induction
        for(int t=T-2;t>=0;t--)
        {
            for(int i=0;i<N;i++)
            {
                ld sum=0;
                for(int j=0;j<N;j++)
                {
                    sum+= a[i][j] * b[j][ob[t+1]] * beta[t+1][j];
                }
                beta[t][i]=sum;
                // cout<<beta[t][i]<<" ";
                // cout<<sum<<" ";
            }
            // cout<<"\n";
        }
        P=0;
        //termination
        for(int i=0;i<N;i++)
        {
            P+=beta[0][i];
            //cout << beta[0][i] << "\n";
        }
        return P;
    }


    //SOLUTION TO 2 PROBLEM
    ld viterbi_algorithm()
    {

        int arg_max=0;
        //step 1 initialisation
        for(int i=0;i<N;i++)
        {
            delta[0][i]=pi[i]*b[i][ob[0]];
            //cout  << delta[0][i] << "\n";
            psi[0][i]=-1;
        }

        //step 2 induction
        for(int t=1;t<T;t++)
        {
            for(int j=0;j<N;j++)
            {
                arg_max=0;
                for(int i=1;i<N;i++)
                {
                    if(delta[t-1][i]*a[i][j] > delta[t-1][arg_max]* a[arg_max][j])
                        arg_max=i;
                }
            // cout<<"here"<<j<<" "<<i<<"\n";
                delta[t][j]=delta[t-1][arg_max]*a[arg_max][j]* b[j][ob[t]];
                psi[t][j]=arg_max;
            }
        }

        arg_max=0;
        for(int i=1;i<N;i++)
        {
            if(delta[T-1][i] > delta[T-1][arg_max])
                arg_max=i;
        }


        // step 3 termination
        pstar=delta[T-1][arg_max];

        //backtrack_qstar(argmax);
        //step 4 back tracking
        qstar[T-1]=arg_max;
        // cout<<"\n";

        for(int t=T-2;t>=0;t--)
        {
            qstar[t]=psi[t+1][(long int)qstar[t+1]];
            // cout<<qstar[t]<<" ";

        }
        for (int i = 0; i < T; i++)
        {
            // cout<<qstar[i]<<" ";
        }


        return pstar;
    }

    void gamma()
    {
        int argmax=0;
        ld devider=0;
        for(int t=0;t<T;t++)
        {
            for(int i=0;i<N;i++)
            {
                devider+=alpha[t][i]*beta[t][i];
            }
            argmax=0;
            for(int i=0;i<N;i++)
            {
                gama[t][i]=alpha[t][i]*beta[t][i]/devider;
                if(gama[t][argmax]<gama[t][i])
                    argmax=i;
            }
            devider=0;
        }

    }

    void zeta_calculation()
    {
        ld devider=0;
        for(int t=0;t<T-1;t++)
        {
            devider=0;
            for(int i=0;i<N;i++)
            {
                for(int j=0;j<N;j++)
                devider+=alpha[t][i]*a[i][j]*b[j][ob[t+1]]*beta[t+1][j];
            }
            for(int i=0;i<N;i++)
            {
                for(int j=0;j<N;j++)
```

```
                zeta[t][i][j]=alpha[t][i]*a[i][j]*b[j][ob[t+1]]*beta[t+1][j]/devider;
            }
        }
    }


    // SOLUTION TO 3 PROBLEM
    void re_estimation()
    {
        ld numerator=0, denominator=0;

        //calculating pi bar
        for(int i=0;i<N;i++)
            pi_bar[i]=gama[0][i];

        //calculating a bar
        for(int i=0;i<N;i++)
        {
            for(int j=0;j<N;j++)
            {
                numerator=0;
                denominator=0;
                for(int t=0;t<T-2;t++)
                {
                    numerator+=zeta[t][i][j];
                    denominator+=gama[t][i];
                }
                a_bar[i][j]=numerator/denominator;
            }
        }

        //calculating b bar
        for(int j=0;j<N;j++)
        {
            for(int k=0;k<M;k++)
            {
                numerator=0;
                denominator=0;
                for(int t=0;t<T;t++)
                {
                    if(ob[t] ==k)
                        numerator+=gama[t][j];
                }
                for(int t=0;t<T-1;t++)
                {
                    denominator+=gama[t][j];
                }
                b_bar[j][k]=max(numerator/denominator , (ld)1e-30 );
            }
        }

    }

    void update()                                // upadting λ
    {
        for(int i=0;i < N;i++) pi[i]=pi_bar[i];  // updating PI

        for(int i=0;i < N;i++)                   // updating A matrix
        {
            for(int j=0;j < N;j++)
            {
                a[i][j]=a_bar[i][j];
            }
        }

        for(int i=0;i < N;i++)                  // upadting B matrix
        {
            for(int j=0;j < M;j++)
            {
                b[i][j]=b_bar[i][j];
            }
        }
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

void recognise()
{
    string word[11]={"yes","no","start","stop","wikipedia",
                    "youtube","search","speech","dollar","corona",
                    "gold"
                    };


        string s1="",s2="";
        int i=1;
        string filename="WORD";
        //LIVE TESTING................................................................................
        while(i<=2)
```

```
{

     data.clear(); //clear data[]
     ob.clear();   //clearing for new file


    string path= "RM\\\\Recording_Module.exe 3 ";
    string filename="WORD";
    path.append(filename);
    path.append(to_string((ld)i));
    path.append(".wav ");
    filename="WORD";
    filename.append(to_string((ld)i));
    filename.append(".txt");
    path.append(filename);

                    int n=path.size()+1;

    char ar[100];

                    strcpy(ar,path.c_str());

   /* for (int i = 0; i < path.size(); i++)
    {
        ar[i]=path[i];
        //  cout<<a[i];
    }*/
    // system("RM\\Recording_Module.exe 3 LIVETEST.wav LIVETEST.txt");
    system(ar);
    // system("taskkill/im Recording_Module.exe");
    string test =filename;
    cout<<filename<<"\n";

    dcshift(test);
    findmax(test);
    findsteadyframes(test);
    ci_cal();
    // adding to universe
    findobservationseq();
    fstream lamda;
    lamda.open("projectlamdas.txt",ios::in|ios::out);
    if(!lamda)cout<<"\nERROR OPENING lamdamodels.txt\n";
    else
    {
        ld maxp=-1,modelno=-1;
        for (int model = 0; model <= 10; model++)
        {
            string val;
            for(int m=0;m<N;m++)
            {
                lamda>>val; pi[m]=stold(val);
            }

            for(int m=0;m<N;m++)
            {
                for(int n=0;n<N;n++)
                {
                    lamda>>val;  a[m][n]=stold(val);
                }
            }


            for(int m=0;m<N;m++)
            {
                for(int n=0;n<M;n++)
                {
                    lamda>>val; b[m][n]=stold(val);
                }
            }


            ld tempp=forward();
                                // cout<<"-->"<<max(tempp,(ld)maxp)<<"\n";
            cout<<"\n FOR "<<model<<" P ="<<tempp;
            if(tempp > maxp)
            {
                maxp=tempp;
                modelno=model;
                // cout<<modelno<<"\n";
            }

                        // cout<<endl;
                        // for(int m=0;m<N;m++)
                        //     { cout<<pi[m]<<" ";}
                        // cout<<"\n";
                        // for(int m=0;m<N;m++)
                        //     {for(int n=0;n<N;n++)
                        //         {cout<<a[m][n]<<" ";}
```

```
                                //         cout<<endl;
                                //     }
                                // for(int m=0;m<N;m++)
                                //     {for(int n=0;n<M;n++)
                                //         {cout<<b[m][n]<<" ";}
                                //             cout<<endl;
                                //     }

                }
                cout<<"\nRECOGNISED WORD --> "<<word[(int)modelno]<<"\n";
                if(i==1)s1=word[(int)modelno];
                if(i==2)s2=word[(int)modelno];
                // if(td==modelno)r++;
                cout<<"\n------------------------------------------\n";
            }
            lamda.close();

            i++;

        }

        cout<<s1<<"\n"<<s2<<"\n";

        ////BROWSER OPENING ////////////////////////////////////////////////////////////
        if(s1=="search")
        {
            string q="start https://google.com/search?q=";
            q.append(s2);
            char ar[100];
            for (int i = 0; i < 100; i++)
            {
                ar[i]=' ';
            }

            for (int i = 0; i < q.size(); i++)
            {
                ar[i]= q[i];
                //  cout<<a[i];
            }
                            if(s2=="corona")
                                     {
                                              system("start https://www.mohfw.gov.in");
                                }
                                else
                                              {
                                                       if(s2=="gold")system("start https://www.goodreturns.in/gold-
rates/");

                                                       else system(ar);
                                    }

        }
        if(s1=="wikipedia")
        {
            string q="start https://en.wikipedia.org/wiki/";
            q.append(s2);
            char ar[100];
            for (int i = 0; i < 100; i++)
            {
                ar[i]=' ';
            }
            for (int i = 0; i < q.size(); i++)
            {
                ar[i]= q[i];
                //  cout<<a[i];
            }
            system(ar);

        }
        if(s1=="youtube")
        {
            string q="start https://www.youtube.com/results?search_query=";
            q.append(s2);
            char ar[100];
            for (int i = 0; i < 100; i++)
            {
                ar[i]=' ';
            }
            for (int i = 0; i < q.size(); i++)
            {
                ar[i]= q[i];
                //  cout<<a[i];
            }
            system(ar);

        }
        ///////////////////////////////////////////////////////////////////////////////////
}
```