```
#include "stdafx.h"
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<string.h>

#define N 5
#define M 32
#define T 150
#define utterrance 20
#define no_of_digits 10
#define test 10
#define p 12
#define frame_size 320

//all global variable declarations
static int qt_star[T]={0};
static double observation_test[T][frame_size+1]={0};
static long double alpha[T+1][N+1]={0};
static long double beta[T+1][N+1]={0};
static int O[utterrance+1][T+1]={0};
static long double A[N+1][N+1]={0};
static long double B[N+1][M+1]={0};
static int Pi[N+1]={0,1,0,0,0,0};
static long double zhi[T+1][N+1][N+1]={0};
static long double gamma[T+1][N+1]={0};
static long double Delta[86][6]={0};
static int Shi[86][6]={0};
static long double avg_A[N+1][N+1]={0};
static long double avg_B[N+1][M+1]={0};
static int avg_Pi[N+1]={0};
static int tO[test+1][T+1]={0};
static int T_values[13][utterrance+1]={0};
static int T_values_test[13][11]={0};
static int testArray[4][4] = {{2,5,7,9},{8,4,7,0},{5,7,6,9},{11,12,-1,-1}};

static double calculate_maximum_amplitude(FILE *fp)
{
        double pmax=0;
        double m=0.0;
        double nmax=0;
        double y=0.0;
        char a[100];
        int i=0;
        while(!feof(fp))
        {
                fgets(a,100,fp);
                y=_atoi64(a);
                if(y>0)
                {
                if(pmax<y)
                        pmax=y;
                }
                else
                {
                        if(nmax>y)
                                nmax=y;
                }

        }
        if(abs(nmax)>abs(pmax))
                m=nmax;
        else
                m=pmax;
```

```
        return abs(m);
}


static double * calculate_RIs(double *test_array,double *Ri)
{

        for(int i=0;i<=p;i++)
        {
                for(int j=1;j<=320-i;j++)
                {
                        Ri[i]+=test_array[j]*test_array[i+j];
                }
        }
        return Ri;
}
//calculating linear predictive co efficients
static double * calculate_ais(double *test_array,double *R,double *a)
{
        double E[13]={0};
        double alpha[13][13]={0};
        double K[13]={0};
        E[0]=R[0];
        for(int i=1;i<=12;i++)
        {
                double sum=0.0;
                for(int j=1;j<=i-1;j++)
                {
                        sum+=alpha[i-1][j]*R[i-j];
                }
                K[i]=(R[i]-sum)/E[i-1];
                alpha[i][i]=K[i];
                for(int j=1;j<=i-1;j++)
                {
                        alpha[i][j]=alpha[i-1][j]-K[i]*alpha[i-1][i-j];
                }
                E[i]=(1-K[i]*K[i])*E[i-1];
        }
        for(int i=1;i<=12;i++)
        {
                a[i]=alpha[12][i];
        }

        return a;
}
//calculating c values
static double * calculate_CIs(double *a,double *R,double *c)
{

        //double c[13];
        double Sigma=R[0]*R[0];
        c[0]=log10(double (Sigma));
        for(int i=1;i<=p;i++)
        {
                double sum=0.0;
                for(int k=1;k<=i-1;k++)
                {
                        sum+=(double(k)/double(i))*c[k]*a[i-k];
                }
                c[i]=a[i]+sum;
        }

        return c;
}
//raised sine function
```

```
static void raised_sine_window(double *w)
{
        for(int i=1;i<=12;i++)
                w[i]=1+6*sin(3.14159265*i/12);
}

//helping function for online testing
static long double testing(int iter,long double MA[N+1][N+1],long double MB[N+1][M+1],int *tO)
{
        //printf("Printing alpha values---------------------------------------------------->\n");
        //initiaisation
        /*
        for(int i=1;i<=iter;i++)
                printf("%d  ",tO[i]);
        printf("\n\n\n");*/
        for(int i=1;i<=N;i++)
        {
                alpha[1][i]=Pi[i]*MB[i][tO[1]];
                //printf("%g \n",MB[i][tO[1][1]]);
                //printf("%d \n",Pi[i]);
        }


        //induction
        for(int t=1;t<=iter-1;t++)
        {
                for(int j=1;j<=N;j++)
                {
                        long double sum=0;
                        for(int i=1;i<=N;i++)
                        {
                                sum+=alpha[t][i]*MA[i][j];
                        }
                        alpha[t+1][j]=sum*MB[j][tO[t+1]];
                }
        }
        /*for(int i=1;i<=N;i++)
        {
                for(int j=1;j<=iter;j++)
                {
                        printf("%e  ",alpha[i][j]);
                }
                printf("\n");
        }*/
        //termination
        long double probability=0;
        for(int i=1;i<=N;i++)
                probability+=alpha[iter][i];

        //printf("Probability = %g \n ",probability);


        //Backward procedure


        //printf("Printing beta values---------------------------------------\n");
        /*for(int j=1;j<=T;j++)
        {
                for(int i=1;i<=N;i++)
                {
                        printf("%e  ",beta[j][i]);
                }
                printf("\n");
        }*/
        return probability;
```

```
}


//for online testing
static int online_testingQ(int question)
{
		char name[500];
		//sprintf(name,"Recording_Module.exe 3 input_file.wav testing.txt");
		//system(name);
		int tc=0;
		FILE *fp_testing_file=fopen("testing.txt","r");
		while(!feof(fp_testing_file))
		{
			int x=0;
			fscanf(fp_testing_file,"%d",&x);
			if(abs(x)>=10)
			{
				while(!feof(fp_testing_file)&&abs(x)>=100)
				{
					fscanf(fp_testing_file,"%d",&x);
					//printf("%d\n",x);
					tc++;
				}
			}
		}

		rewind(fp_testing_file);
		float max_amp=calculate_maximum_amplitude(fp_testing_file);
		rewind(fp_testing_file);
		long double *test_word;
		test_word=(long double *)malloc( (tc+1) * sizeof(long double));
		for(int i=1;i<=tc;i++)
		{
			long double x=0;
			fscanf(fp_testing_file,"%lf",&x);
			test_word[i]=x*(5000/max_amp);

		}
		int t=0;
		double max_energy=0;
		int max_energy_index=0;
		double observation[T+1][frame_size+1]={0};
		int frame_shift=1;
		for(int i=1;i<=T;i++)
		{
			double energy=0;
			for(int j=1;j<=frame_size;j++)
			{
				energy+=test_word[frame_shift]*test_word[frame_shift];
				frame_shift++;
			}
			if(max_energy<energy)
			{
				max_energy=energy;
				max_energy_index=frame_shift;
			}
			if(frame_shift-80+frame_size>tc)
			{
				t=i;
				break;

			}
			else
				frame_shift-=80;
		}
```

```
int frame_s=frame_shift-39;
for(int i=1;i<=90;i++)
{
        for(int j=1;j<=frame_size;j++)
        {
                observation[i][j]=test_word[frame_s];
                frame_s++;
        }
        if(frame_s-80+frame_size>60)
        {
                t=i;
                break;

        }
        else
                frame_s-=80;
}
fclose(fp_testing_file);

FILE *fp_test_observation=fopen("testobservation.txt","w");
double w[p+1]={0};
raised_sine_window(w);

double Ri[T+1][p+1]={0};
for(int i=1;i<=t;i++)
{
        calculate_RIs(observation[i],Ri[i]);
}


double Ai[T+1][p+1]={0};
for(int i=1;i<=t;i++)
{
        calculate_ais(observation[i],Ri[i],Ai[i]);
}



double Ci[T+1][p+1]={0};
for(int i=1;i<=t;i++)
{
        calculate_CIs(Ai[i],Ri[i],Ci[i]);
}

for(int i=1;i<=t;i++)
{
        for(int j=1;j<=p;j++)
        {
                Ci[i][j]=Ci[i][j]*w[j];
        }

}

FILE *fp_codebook=fopen("codebook11.txt","r");
double codebook[M+1][p+1];
for(int i=1;i<=M;i++)
{
        for(int j=1;j<=p;j++)
        {
                double y=0;
                fscanf(fp_codebook,"%lf",&y);
                codebook[i][j]=y;
        }
}
```

```
                double weight[13]={0,1.0,3.0,7.0,13.0,19.0,22.0,25.0,33.0,42.0,50.0,65.0,61.0};
                double dist=1000;
                for(int frame=1;frame<=t;frame++)
                {
                        dist=1000;
                        int index=1;

                        for(int k=1;k<=M;k++)
                        {
                        long double dist1=0;
                        for(int j=1;j<=p;j++)
                        {
                                dist1+=weight[j]*(Ci[frame][j]-codebook[k][j])*(Ci[frame][j]-
codebook[k][j]);

                        }
                        if(dist1<dist)
                        {
                                index=k;
                                dist=dist1;
                        }

                        }


                        fprintf(fp_test_observation,"%d   ",index);
                }
        fprintf(fp_test_observation,"\n");
        fclose(fp_test_observation);
                //testing---------------------------
                //storing models---------------------
        long double MA[no_of_digits][N+1][N+1]={0};
        long double MB[no_of_digits][N+1][M+1]={0};

        int k;
        int q = 0;
        for(int z=0;z<=3;z++)
        {
                k = testArray[question-1][z];  //2,5,7,9
                if(k != -1){
                        q++;
                        char fname_A[100];
                        char fname_B[100];
                        sprintf(fname_A,"A_%d.txt",k);
                        sprintf(fname_B,"B_%d.txt",k);
                        FILE *f_MA=fopen(fname_A,"r");
                        FILE *f_MB=fopen(fname_B,"r");
                        for(int i=1;i<=N;i++)
                        {
                                for(int j=1;j<=N;j++)
                                {
                                        long double temp=0;
                                        fscanf(f_MA,"%lf  ",&temp);
                                        MA[q][i][j]=temp;  //q wala matrix
                                }
                        }
                        for(int i=1;i<=N;i++)
                        {
                                for(int j=1;j<=M;j++)
                                {
                                        long double temp=0;
                                        fscanf(f_MB,"%lf  ",&temp);
                                        MB[q][i][j]=temp;  //q wala matrix
                                }
                        }
```

```
                        fclose(f_MA);
                        fclose(f_MB);
                }
        }
        FILE *ft=fopen("testobservation.txt","r");
        int *test_ob;
        test_ob=(int *)malloc( (t+1) * sizeof(int));
        for(int i=1;i<=t;i++)
        {
                int x=0;
                fscanf(ft,"%d",&x);
                test_ob[i]=x;
        }
                long double prob=0;
                int rec=2;
                for(int k=0;k<=q;k++)  // comparing with each digit : no_of_digits-1
                {

                        long double p1=0;
                        p1=testing(t,MA[k],MB[k],test_ob);
                        printf("Probability of being digit %d =",k);
                        printf("%g\n  ",p1);
                        if(p1>prob)
                        {
                                prob=p1;
                                rec=k;

                        }
                }
                printf("Recognised digit is = %d\n",rec);
                if(rec==-1)
                        printf("Sorry!! coudn't recognise your voice\n");
                return rec;
}
```