/\*

Class-1 Problem-1

Heat Equation solution using Bendre-Schimdt Scheme

\*/

#include<iostream>

#include<stdio.h>

using namespace std;

int main()

{

double h=1.0,k=0.0125,v,c=4.0;

double X=8.0,T=0.05,x,t;

int a,b;

v=(c\*c\*k)/(h\*h);

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++)

{

x=h\*i;

u[i][0]=4\*x-x\*x/2.0;

}

//Boundary Condition

for(int i=1; i<b; i++)

u[0][i]=u[a-1][i]=0;

for(int j=0; j<b-1; j++)

{

for(int i=1; i<a-1; i++)

{

u[i][j+1]=v\*u[i+1][j]+(1.0-2\*v)\*u[i][j]+v\*u[i-1][j];

}

}

cout<<">---Time Varying--->--->--->\n";

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.4f ",u[i][j]);

}

printf("\n\n");

}}

/\*

Class-1 Problem-2

\*/

#include<iostream>

#include<stdio.h>

using namespace std;

int main()

{

double h=1.0,k=0.0125,v,c=4.0;

double X=5.0,T=0.05,x,t;

int a,b;

v=(c\*c\*k)/(h\*h);

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++)

{

x=h\*i;

u[i][0]=x\*x\*(5.0-x);

}

//Boundary Condition

for(int i=1; i<b-1; i++)

u[0][i]=u[a-1][i]=0;

//Assuming u[i][-1]=0

for(int i=1; i<a-1; i++)

u[i][1]=u[i-1][0]+u[i+1][0]-0;//assuming this as initial condition and zero for u[i][-1]

for(int j=1; j<b-1; j++)

for(int i=1; i<a-1; i++)

u[i][j+1]=u[i-1][j]+u[i+1][j]-u[i][j-1];

cout<<">---Time Varying--->--->--->\n";

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.4f ",u[i][j]);

}

printf("\n\n");

}

}

/\*

Class-2 Problem-1

Implement Lax Method to solve 1-D Wave Equation

\*/

#include<iostream>

#include<stdio.h>

#include<cmath>

#define PI 3.14

using namespace std;

int main()

{

double h=1.0,k=0.6,v;

double X=20.0,T=10.0,x;

int a,b;

int n=3;//given in problem

v=k/h;

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++)

{

x=h\*i;

u[i][0]=sin(2.0\*n\*PI\*(x/10.0));

}

//Boundary Condition

for(int i=1; i<b; i++)

u[0][i]=u[a-1][i]=0;

for(int j=0; j<b-1; j++)

for(int i=1; i<a-1; i++)

u[i][j+1]=(u[i+1][j]+u[i-1][j])/2.0 - v\*(u[i+1][j]-u[i-1][j])/2.0;

cout<<">---Time Varying--->--->--->\n";

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.2f ",u[i][j]);

}

printf("\n\n");

}

}

/\*

Class-2 Problem-2

Implement Lax-Wendoff Method to solve 1-D Wave Equation

\*/

#include<iostream>

#include<stdio.h>

#include<cmath>

#define PI 3.14

using namespace std;

int main()

{

double h=1.0,k=0.6,v,c=4.0;

double X=20.0,T=10.0,x;

int a,b;

int n=3;//given in problem

v=k/h;

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++)

{

x=h\*i;

u[i][0]=sin(2.0\*n\*PI\*(x/10.0));

}

//Boundary Condition

for(int i=1; i<b; i++)

u[0][i]=u[a-1][i]=0;

for(int j=0; j<b-1; j++)

for(int i=1; i<a-1; i++)

u[i][j+1]=u[i][j] - v\*(u[i+1][j]-u[i-1][j])/2.0 +v\*v\*(u[i+1][j]-2.0\*u[i][j]+u[i-1][j])/2.0;

cout<<">---Time Varying--->--->--->\n";

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.2f ",u[i][j]);

}

printf("\n\n");

}

}

/\*

Class-3 Method-1

Solving Heat Equation

\*/

#include<iostream>

#include<stdio.h>

#include<cmath>

using namespace std;

int main()

{

double h=0.1,k=0.1,v,c=10;

double X=1.0,T=1.0,x;//taking any random value of T

int a,b;

v=k/(h\*h);

v/=c;//otherwise solution overflows

cout<<v<<"\n";

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++)

u[i][0]=0;

//Boundary Condition

for(int i=1; i<b; i++)

{

u[0][i]=0;

u[a-1][i]=k\*i;

}

for(int j=0; j<b-1; j++)

for(int i=1; i<a-1; i++)

u[i][j+1]=(double)u[i][j] + v\*(u[i+1][j]-2.0\*u[i][j]+u[i-1][j]);

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.2f ",u[i][j]);

}

printf("\n\n");

}

}

/\*

Class-3 Method-1

Solving Heat Equation

\*/

#include<iostream>

#include<stdio.h>

#include<cmath>

using namespace std;

int main()

{

double h=0.1,k=0.1,v,c=10;

double X=1.0,T=1.0,x;//taking any random value of T

int a,b;

v=k/(h\*h);

v/=c;//otherwise solution overflows

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++)

u[i][0]=0;

//Boundary Condition

for(int i=1; i<b; i++)

{

u[0][i]=0;

u[a-1][i]=k\*i;

}

for(int i=1,j=1;i<a-1;i++)

u[i][j]=(2.0\*v\*(u[i+1][j-1]+u[i-1][j-1]))/(1.0+2.0\*v);//assuming u[i][-1]=0

for(int j=1; j<b-1; j++)

for(int i=1; i<a-1; i++)

u[i][j+1]=(u[i][j-1] + 2.0\*v\*(u[i+1][j] - u[i][j-1] + u[i-1][j]))/(1.0+2.0\*v);

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.2f ",u[i][j]);

}

printf("\n\n");

}

}

/\*

Class-3

\*/

#include<iostream>

#include<stdio.h>

#include<cmath>

#define PI 3.14

using namespace std;

int main()

{

double h=0.1,k=0.1,v,c=0.2;

double X=1.0,T=1.0,x;//taking any random value of T

int a,b;

v=k/(h\*h);

v\*=c;

v/=10;//otherwise solution will overflow so diving with random number

a=X/h+1;

b=T/k+1;

double u[a][b];

for(int i=0; i<a; i++)

for(int j=0; j<b; j++)

u[i][j]=0;

//Initial Condition

for(int i=0; i<a; i++){

x=i\*h;

u[i][0]=100.0\*sin(PI\*x);

}

//Boundary Condition

for(int i=1; i<b; i++)

u[0][i]=u[a-1][i]=0;

for(int j=0; j<b-1; j++)

for(int i=1; i<a-1; i++)

u[i][j+1]=u[i][j] + v\*(u[i+1][j] - 2.0\*u[i][j] + u[i+1][j]);

for(int i=0; i<a; i++)

{

for(int j=0; j<b; j++)

{

printf("%.2f ",u[i][j]);

}

printf("\n\n");

}

}

**Chinese Remainder Theorem**

#include<bits/stdc++.h>

#define ll long long

#define MAX 1000005

using namespace std;

vector<int> prime;

bool b[MAX];

ll power(ll a,ll n,ll m)

{

if(n==0)

return 1;

else

{

ll v=power(a,n/2,m);

v=v\*v%m;

if(n&1)

v=v\*a%m;

return v;

}

}

ll gcd(ll a,ll b)

{

if(b==0)

return a;

if(a==0)

return b;

else

return gcd(b,a%b);

}

ll phi(ll n)

{

for(int i=0;i<prime.size() && prime[i]<=n;i++)

if(n%prime[i]==0)

n-=n/prime[i];

return n;

}

int main()

{

int i,j,n;

memset(b,true,sizeof b);

for(i=2;i\*i<MAX;i++)

{

if(b[i])

{

prime.push\_back(i);

for(j=i\*i;j<MAX;j+=i)

b[j]=false;

}

}

for(i;i<MAX;i++)

if(b[i])

prime.push\_back(i);

cout<<"Number of equation \n";

cin>>n;

ll a[n],m[n];

bool f=true;

ll lcm=1,g;

for(i=0;i<n;i++)

{

printf("remainder in equation %d and modulo \n",i+1);

cin>>a[i]>>m[i];

if(gcd(a[i],m[i])!=1)

f=false;

g=gcd(lcm,m[i]);

if(g!=1)f=false;

lcm=lcm\*m[i];

}

if(!f)

cout<<"Check input Error\n";

else

{

ll ans=0;

for(i=0;i<n;i++)

{

ll help=lcm/m[i];

ans+=((a[i]\*help%lcm)\*power(help%m[i],phi(m[i])-1,m[i]))%lcm;

if(ans>=lcm)ans-=lcm;

}

cout<<"ans=="<<ans<<"\n";

}

}

**Euler Theorem**

#include<bits/stdc++.h>

#define MAX 100005

using namespace std;

vector<int> prime;

bool b[MAX];

int phi(int n)

{

for(int i=0;i<prime.size() && prime[i]<=n;i++)

if(n%prime[i]==0)

n-=n/prime[i];

return n;

}

int main()

{

int i,j,n;

memset(b,true,sizeof b);

for(i=2;i\*i<MAX;i++)

{

if(b[i])

{

prime.push\_back(i);

for(j=i\*i;j<MAX;j+=i)

b[j]=false;

}

}

for(i;i<MAX;i++)

if(b[i])

prime.push\_back(i);

cin>>n;

cout<<phi(n);

}

**RSA**

#include<bits/stdc++.h>

#define MAX 1000005

#define ll long long

using namespace std;

vector<int> prime;

bool b[MAX];

int phi(int n)

{

for(int i=0;i<prime.size() && prime[i]<=n;i++)

if(n%prime[i]==0)

n-=n/prime[i];

return n;

}

ll power(ll a,ll n,ll m)

{

if(n==0)

return 1;

else

{

ll v=power(a,n/2,m);

v=v\*v%m;

if(n&1)

v=v\*a%m;

return v;

}

}

ll gcd(ll a,ll b)

{

if(b==0)

return a;

if(a==0)

return b;

else

return gcd(b,a%b);

}

void RSA(ll n,ll d)

{

ll p=-1,q=1;

int i;

for(i=0;i<prime.size() && prime[i]<=n ;i++)

{

if(n%prime[i]==0)

{

p=prime[i];

q=n/p;

break;

}

}

ll phin=(p-1)\*(q-1);

if(p==-1 || gcd(phin,d)!=1)

{

cout<<"Error in input\n";

return;

}

else

{

cout<<"key is\t"<<power(d,phi(phin)-1,phin);

}

}

int main()

{

int i,j;

memset(b,true,sizeof b);

for(i=2;i\*i<MAX;i++)

{

if(b[i])

{

prime.push\_back(i);

for(j=i\*i;j<MAX;j+=i)

b[j]=false;

}

}

for(i;i<MAX;i++)

if(b[i])

prime.push\_back(i);

ll n,d;

cout<<"Enter n and d\n";

cin>>n>>d;

RSA(n,d);

}

**Affine Cipher**

#include<iostream>

#include<stdio.h>

#include<vector>

using namespace std;

//Encryption function

void encrypt(string plain){

vector<int> inputnumbers,outputnumbers;

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

inputnumbers.push\_back(plain[i]-'a');

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

outputnumbers.push\_back((inputnumbers[i]\*5)+6);

if(outputnumbers[i]>=26)

outputnumbers[i]%=26;

}

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

cout<<char(outputnumbers[i]+65);

}

//Decryption function

void decrypt(string cipher){

vector<int> inputnumbers,outputnumbers;

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

inputnumbers.push\_back((cipher[i]-'A'));

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

int temp=inputnumbers[i]-6;

if(temp<0)

temp+=26;

outputnumbers.push\_back((temp\*21)%26);

}

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

cout<<char(outputnumbers[i]+97);

}

int main(){

//string input;//For taking custom string as input

//cin>>input;

decrypt("XPALASXYFGUKPXUSOGEUTICCDGFXANMGNVS");

cout<<endl;

encrypt("thebestofaightismakinquupafterwards");

}

**Aditya Kapoor**

**12412EN004**

**Mathematics and Computing Part IV**

**Dept. Of Mathematical Sciences**

**Mathematics lab 2**

**Sessional FIle**