DEPARTMENT OF COMPUTER TECHNOLOGY

SKILL-2 B.Sc. CT : NETORK SECURITY LAB

VI SEMESTER

PROGRAMME OUTCOME

PO - 1 The Programme exhibits solid fundamental knowledge in mathematics, computer science with a breadth of inter-disciplinary knowledge.

PO - 2 Ability for independent thinking, possesses problem-solving skills, and excels in the capability for self-learning to allow for life-long learning.

PO - 3 Capability to apply design and development principles in the construction of software systems of varying complexity.

PO - 4 Developing leadership qualities, and good communication, teamwork, social, and professional skills.

PO - 5 An ability to design, implement and evaluate a computational system to meet desired needs within realistic constraints.

PO - 6 An ability to use appropriate techniques, skills, and tools necessary for computing practice.

PO - 7 Recognition of the need for and an ability to engage in continuing professional development

PO - 8 Ability to apply knowledge of computing, mathematics and business accounting principles appropriate to the discipline.

PO - 9 Students will be able to communicate effectively in a variety of professional contexts

PO - 10 Identify, formulate, and solve computer science problems (problem formulation)

Course Objectives

- ✓ To understand the Modes of data transmission
- ✓ To understand the Transmission Errors: Detection and correction
- ✓ To understand the various routing algorithms.
- ✓ To know the concept of data transfer between nodes
- ✓ To know the concept of inter process communication technique that is used for client-server based applications

List of Experiments

1. Write a program to encrypt the data using the encryption methods: (i) Substitution Ciphers (ii) Transposition Ciphers

2. Write a program to implement DES algorithm.

3. Write a program to implement the Public Key Cryptography using Diffie - Hellman Algorithm.

4. Write a program to implement the Public Key Cryptography using RSA algorithm.

5. Write a program to secure the Database using User Authentication Security.

6. Write a server security program for Dynamic Page Generation.

Prerequisite

Knowledge about simple networking concepts

Software requirement

- ✓ Turbo C
- ✓ Java- JDK

Course Outcome

After completing the course, students will be able to:

- ✓ **Understand the structure and organization of computer networks**
- ✓ **Understand the error detection and correction mechanism**
- ✓ **Understand the concept of routing algorithms in networks**
- ✓ **Understand the concept of different communication modes**
- ✓ **Understand the concept of file transfer between client and server**
- ✓ **Understand the concept of Remote process communication**

Introduction to Socket Programming and methods used in Socket

Socket Programming

Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.

When the connection is made, the server creates a socket object on its end of the communication. The client and the server can now communicate by writing to and reading from the socket.

The java.net.Socket class represents a socket, and the java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.

The following steps occur when establishing a TCP connection between two computers using sockets −

- The server instantiates a **ServerSocket** object, denoting which port number communication is to occur on.
- The server invokes the **accept()** method of the **ServerSocket** class. This method waits until a client connects to the server on the given port.
- After the server is waiting, a client instantiates a Socket object, specifying the server name and the port number to connect to.
- The constructor of the Socket class attempts to connect the client to the specified server and the port number. If communication is established, the client now has a Socket object capable of communicating with the server.
- On the server side, the accept() method returns a reference to a new socket on the server that is connected to the client's socket.

After the connections are established, communication can occur using I/O streams. Each socket has both an **OutputStream** and an **InputStream**. The client's OutputStream is connected to the server's InputStream, and the client's InputStream is connected to the server's OutputStream.

TCP is a two-way communication protocol, hence data can be sent across both streams at the same time. Following are the useful classes providing complete set of methods to implement sockets.

ServerSocket Class Methods

The **java.net.ServerSocket** class is used by server applications to obtain a port and listen for client requests.

The ServerSocket class has four constructors −

S.No.	Method & Description
1	**public ServerSocket(int port) throws IOException** Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
2	**public ServerSocket(int port, int backlog) throws IOException** Similar to the previous constructor, the backlog parameter specifies how many incoming clients to store in a wait queue.
3	**public ServerSocket(int port, int backlog, InetAddress address) throws IOException** Similar to the previous constructor, the InetAddress parameter specifies the local IP address to bind to. The InetAddress is used for servers that may have multiple IP addresses, allowing the server to specify which of its IP addresses to accept client requests on.
4	**public ServerSocket() throws IOException** Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket.

If the ServerSocket constructor does not throw an exception, it means that your application has successfully bound to the specified port and is ready for client requests.

Following are some of the common methods of the ServerSocket class −

Sr.No.	Method & Description
1	**public int getLocalPort()** Returns the port that the server socket is listening on. This method is useful if you passed in 0 as the port number in a constructor and let the server find a port for you.
2	**public void bind(SocketAddress host, int backlog)** Binds the socket to the specified server and port in the SocketAddress object. Use this method if you have instantiated the ServerSocket using the no-argument constructor.

When the ServerSocket invokes accept(), the method does not return until a client connects. After a client does connect, the ServerSocket creates a new Socket on an unspecified port and returns a reference to this new Socket. A TCP connection now exists between the client and the server, and communication can begin.

Socket Class Methods

The **java.net.Socket** class represents the socket that both the client and the server use to communicate with each other. The client obtains a Socket object by instantiating one, whereas the server obtains a Socket object from the return value of the accept() method.

The Socket class has five constructors that a client uses to connect to a server −

Sr.No.	Method & Description
1	**public Socket(String host, int port) throws UnknownHostException, IOException.** This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
2	**public Socket(InetAddress host, int port) throws IOException** This method is identical to the previous constructor, except that the host is denoted by an InetAddress object.
3	**public Socket(String host, int port, InetAddress localAddress, int localPort) throws IOException.** Connects to the specified host and port, creating a socket on the local host at the specified address and port.

When the Socket constructor returns, it does not simply instantiate a Socket object but it actually attempts to connect to the specified server and port.

Some methods of interest in the Socket class are listed here. Notice that both the client and the server have a Socket object, so these methods can be invoked by both the client and the server.

Sr.No.	Method & Description
1	**public void connect(SocketAddress host, int timeout) throws IOException** This method connects the socket to the specified host. This method is needed only when you instantiate the Socket using the no-argument constructor.
2	**public InetAddress getInetAddress()** This method returns the address of the other computer that this socket is connected to.
3	**public int getPort()** Returns the port the socket is bound to on the remote machine.
4	**public int getLocalPort()** Returns the port the socket is bound to on the local machine.
5	**public SocketAddress getRemoteSocketAddress()** Returns the address of the remote socket.
6	**public InputStream getInputStream() throws IOException** Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
7	**public OutputStream getOutputStream() throws IOException** Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket.
8	**public void close() throws IOException** Closes the socket, which makes this Socket object no longer capable of connecting again to any server.

InetAddress Class Methods

This class represents an Internet Protocol (IP) address. Here are following usefull methods which you would need while doing socket programming −

Sr.No.	Method & Description
1	**static InetAddress getByAddress(byte[] addr)** Returns an InetAddress object given the raw IP address.
2	**static InetAddress getByAddress(String host, byte[] addr)** Creates an InetAddress based on the provided host name and IP address.
3	**static InetAddress getByName(String host)** Determines the IP address of a host, given the host's name.
4	**String getHostAddress()** Returns the IP address string in textual presentation.
5	**String getHostName()** Gets the host name for this IP address.
6	**static InetAddress InetAddress getLocalHost()** Returns the local host.
7	**String toString()** Converts this IP address to a String.

Ex No: 01 Program To Encrypt the data using the encryption methods: i) Substitution Ciphers (ii) Transposition Ciphers

AIM:

Write a program to encrypt the data using the encryption methods: (i) Substitution Ciphers (ii) Transposition Ciphers

Procedure to run a program

Step 1: Locate the TC.exe file and open it. You will find it at location **C:\TC\BIN\.**

Step 2: File > New (as shown in above picture) and then write your C program

Step 3: Save the program using F2 (OR file > Save), the extension should be **".c"**.

Step 4: Compile the program using Alt + F9 **OR** Compile > Compile

Step 5: Press Ctrl + F9 to Run (or select Run > Run in menu bar) the C program.

Step 6: Alt+F5 to view the output of the program at the output screen.

SOURCE CODE:

SUBSITUTION CIPHERS

The **substitution** and **transposition** techniques are used for converting a plaintext into ciphertext,where **substitution** technique replaces the characters
whereas **transposition** technique rearranges the characters to form a ciphertext.

PROGRAM:

```
#include<iostream.h>   // Header file declaration.
#include<string.h>
#include<time.h>
#include<fstream.h>
#include<conio.h>
void encrypt(void);  // Method Declaration
void decrypt(void);
int main()
{
int choice;  // Variable declaration
while(choice!=3)
{
cout<<"what operation would you like to perform? \n";
cout<<"1-encrypt\n";
cout<<"2-Decrypt\n";
cout<<"3-quit\n";
cin>>choice;
clrscr();
switch (choice)
{
case 1:
  encrypt(); //Perform encryption  operation
  break;
case 2:
  decrypt(); //Perform Decryption operation
  break;
default:
cout<<"please enter a valid response:\n";
break;
}
}
return 0;
}
void encrypt()
{
```

```
char string,cipher; //Declare the variable to perform  encryption
ofstream fout;
/*string plaintext;
string ciphertext;
string cipherletter;
string shift_string_1; //
string shift_string_2;
*/
char plaintext[10];
char ciphertext[10];
char cipherletter;
char shift_string_1[10];
char shift_string_2[10];
char shift_letter;
signed int cipher_num;
signed int cipher_num_2;
signed int shifter; //Variable declaration
signed int shift_num;
char letter;
cout<<"Enter your word here:\n";
cin>>plaintext;
signed int slength=strlen(plaintext);  // Assign the variable for plain Text
signed int loops=0;
signed int seed;
fout.open("key.txt"); //open the key.txt file
fout.close(); //close the file
fout.open("key.txt",ios::app);
fout<<"PLAINTEXT:"<<plaintext<<endl;
fout<<"key:";
while(loops<slength) //check the condition
{
shifter=loops;
if(loops==0)
{
shifter=slength;
}
else if(loops%2<1)
{
shifter= shifter*-1; //perform the encryption operation
}
else
{
shifter=loops;
}
fout<<shifter;
cipher_num=int(plaintext[loops]);
cipher_num_2=cipher_num+shifter; //increment the cipher text variable
letter=char(cipher_num_2);
cipherletter=letter;
//ciphertext.append(cipherletter);
//ciphertext[slength]=cipher letter;
loops++;
}
fout<<endl<<"ciphertext:"<<ciphertext<<endl; //print the cipher text
fout.close();
getch();
}
void decrypt()
{
char encrypted[10]; //variable declaration
char key[10]; //key value declaration
char deciphered;
cout<<"please enter the encrypted text:\n";
cin>>encrypted; //Get the input variable for encryption
clrscr();
cout<<"please enter the key,without signs:\n";
cin>>key;//Get the key input value
signed short int shifter;
char letter;
char new_letter;
int cipher_num;
int new_num;
char decipher[10];
char holder;
int slength;
int shifted_letter_num;
slength=strlen(encrypted); //Assign the encrypted string length
int loops=0;
int temp_num;
int loop1=0;
while(loops<slength) //Check the condition with  string length
{
temp_num=key[loops]; //Assign the key value to temp variable
cipher_num=temp_num-48; //Decrement the temp variable
cout<<cipher_num; //Print the cipher text
if(loops % 2<1)
{
cipher_num=cipher_num*-1;//Decrement the cipher text value
}
shifted_letter_num=int(encrypted[loops]); //Assign encrypted value to Shift num
cout<<"ASCII of the encrypted text:"<<shifted_letter_num<<endl;
new_num=shifted_letter_num-cipher_num; //Decrement the cipher_num and store it in
new_num
letter=char(new_num);
holder=letter;
cout<<"Decrypted letter:"<<holder<<endl;
decipher[loop1]=holder;
loops++; //increment the loop
}
cout<<"Decrypt done.Decrypted text:"<<decipher<<endl; //Print the decrypted
```

```
getch();
}
```

SUBSTITUTION CIPHER

```
Enter the Key: 1 2 3
Enter the String: CIPHER
IOVNKX
```

1(B).TRANSPOSITION CIPHERS

```
#include<iostream.h>  // Header file declaration.
#include<conio.h>
void main()
{
clrscr();
int a,b,c,z;
for(z=0;z<10;z++) //perform the loop operation
{
c=1;
c+=z;
b=0;
for(a=1;a<11;a++)
{
c*=a;
b+=c;
cout<<b<<" "; //Print the Transposition cipher
}
cout<<endl;
}
getch();
}
```

TRANSPOSITION CIPHER

```
Enter the String: TRANSPOSITION CIPHER
Enter the Crypt Pass: 34251
34251
TRANS
```

```
POSIT
ION C
IPHER
Encrypted String: STCRASNHTPIIROOPNI E
Decrypted String: TRANSPOSITION CIPHER
```

Ex No:02 Write a program to implement DES algorithm.

AIM:

Write a program to implement DES algorithm.

DES ALGORITHM:

- The Data Encryption Standard (**DES**) is an outdated symmetric-key method of data encryption. ...
- It was the first encryption **algorithm** approved by the U.S. government for public disclosure.
- This ensured that **DES** was quickly adopted by industries such as financial services, where the need for strong encryption is high.

Procedure to run a program

Step 1: Locate the TC.exe file and open it. You will find it at location **C:\TC\BIN\.**

Step 2: File > New (as shown in above picture) and then write your C program

Step 3: Save the program using F2 (OR file > Save), the extension should be **".c**".

Step 4: Compile the program using Alt + F9 **OR** Compile > Compile

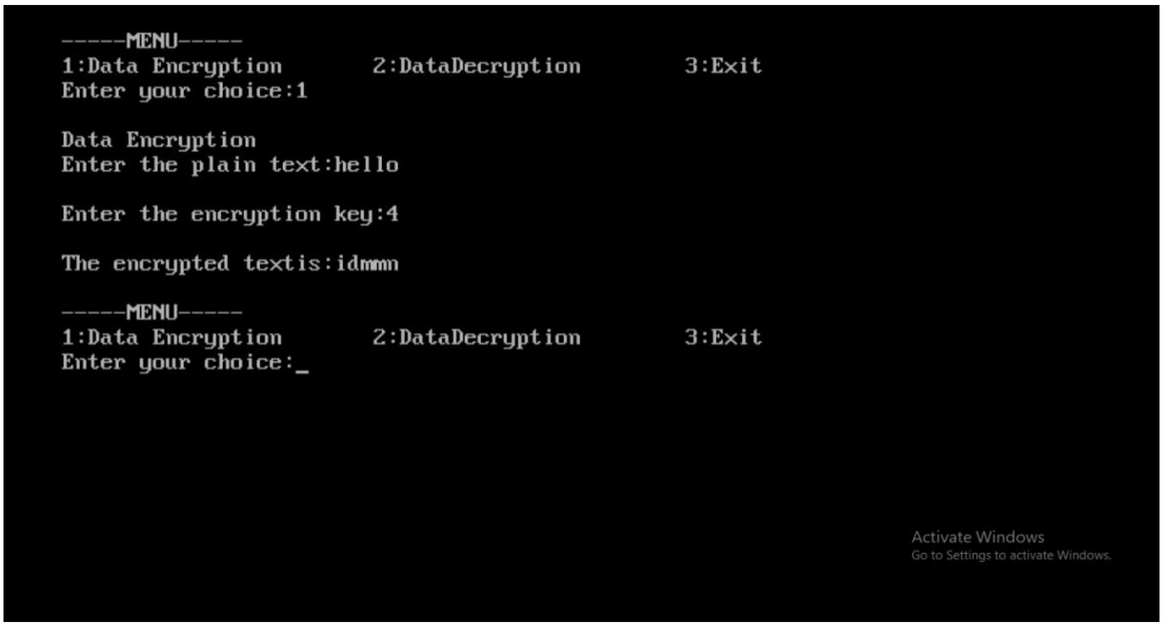
Step 5: Press Ctrl + F9 to Run (or select Run > Run in menu bar) the C program.

Step 6: Alt+F5 to view the output of the program at the output screen.

<u>SOURCE CODE:</u>

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<string.h>
void main()
{
int i,ch,lp;
char cipher[50],plain[50];
char key[50];
clrscr();
while(1)
{
printf(" \n-----MENU-----");
printf("\n1:Data Encryption\t2:DataDecryption\t3:Exit");
printf("\nEnter your choice:");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\nData Encryption");
printf("\nEnter the plain text:");
fflush(stdin);  //clear the previous input buffer//
scanf("%s",plain);  //gets plain text//
printf("\nEnter the encryption key:");
scanf("%s",key);  //gets encryption key//
lp=strlen(key); //get length of key if key=hello then lp=5//
for(i=0;plain[i]!='\0';i++) //"loop runs untill every key gets
encrypted" '\0' indicates end of array //
cipher[i]=plain[i]^lp;  // (STEP 1)^ or xor operator //
cipher[i]='\0';  //indicates  end of array //
printf("\nThe encrypted textis:");
puts(cipher);   //encrypted text is printed//
break;
case 2:
printf("\nData decryption");
for(i=0;cipher[i]!='\0';i++)
plain[i]=cipher[i]^lp;    //reverse of STEP 1 . Decrtyption is done//
printf("\nDecrypted text is:");
puts(plain); //print Decrypted text//
break;
case 3:
exit(0); //system exit//
}
}
getch();
}
```

OUTPUT :



```
------MENU------
1:Data Encryption       2:DataDecryption        3:Exit
Enter your choice:1

Data Encryption
Enter the plain text:hello

Enter the encryption key:4

The encrypted textis:idmmn

------MENU------
1:Data Encryption       2:DataDecryption        3:Exit
Enter your choice:_
```

Ex No :03 Write a program to implement the Public Key Cryptography using Diffie -Hellman Algorithm.

AIM:

Write a program to implement the Public Key Cryptography using Diffie -Hellman Algorithm.

Procedure to run a Program

Step 1: Write a server and client programs on the notepad and save it with **.java** extension,

Step 2: Open two different Command Prompt.

Step 3: Set the directory in which the .java file is saved.

Step 4: Use the following command to compile the Java programs. It generates a .class file in the same folder. It also shows an error if any.

SOURCE CODE:

```
import java.util.*;
import java.math.BigInteger; ;// to calculate the Arithmetic

sum of two BigIntegers.//

public class DiffieHellmanBigInt {

        final static BigInteger one = new BigInteger("1");

        public static void main(String args[]) {
                Scanner stdin = new Scanner(System.in);
                BigInteger p;

                // Get a start spot to pick a prime from the
user.//
                System.out.println("Enter the approximate value of
p you want.");
                String ans = stdin.next();
                p = getNextPrime(ans);
                System.out.println("Your prime is "+p+".");

                // Get the base for exponentiation from the
user//
                System.out.println("Now, enter a number in
between 2 and p-1.");
                BigInteger g = new BigInteger(stdin.next());

                // Get A's secret number//
                System.out.println("Person A: enter your secret
number now.");
                BigInteger a = new BigInteger(stdin.next());

                // Make A's calculation//
                BigInteger resulta = g.modPow(a,p);

                // This is the value that will get sent from A to
B//
                // This value does NOT compromise the value of
a easily//
                System.out.println("Person A sends to person B
"+resulta+".");
```

```
                // Get B's secret number//
                System.out.println("Person B: enter your secret
number now.");
                BigInteger b = new BigInteger(stdin.next());

                // Make B's calculation//
                BigInteger resultb = g.modPow(b,p);

                // This is the value that will get sent from B to
A//
                // This value does NOT compromise the value of
b easily//
                System.out.println("Person B sends to person A
"+resultb+".");

                // Once A and B receive their values, they make
their new calculations//
                // This involved getting their new numbers and
raising them to the
                // same power as before, their secret number//
                BigInteger KeyACalculates =
resultb.modPow(a,p);
                BigInteger KeyBCalculates =
resulta.modPow(b,p);

                // Print out the Key A calculates//
                System.out.println("A takes "+resultb+" raises it to
the power "+a+" mod "+p);
                System.out.println("The Key A calculates is
"+KeyACalculates+".");

                // Print out the Key B calculates//
```

```
		System.out.println("B takes "+resulta+" raises it to
the power "+b+" mod "+p);
		System.out.println("The Key B calculates is
"+KeyBCalculates+".");

	}

	public static BigInteger getNextPrime(String ans) {

		BigInteger test = new BigInteger(ans);
		while (!test.isProbablePrime(99))
			test = test.add(one);
		return test;
	}

}
```

Ex No : 04 **To implement the Public Key Cryptography using RSA algorithm.**

AIM: To implement the Public Key Cryptography using RSA algorithm.

RSA Algorithm:

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption.

The RSA algorithm holds the following features

Step 1: **Generate the RSA modulus**

The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown

n=p*q

Step 2: **Derived Number (e)**

Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

Step 3: **Public key**

The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

Step 4: **Private Key**

Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows

ed = 1 mod (p-1) (q-1)
Encryption Formula

Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −

Cipher text = P^e mod n
Decryption Formula

The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as **Plaintext = C^d mod n**

```
#include<stdio.h> // Header file for  declaration
#include<conio.h>
#include<ctype.h>
#include<math.h>
#include<string.h>
void main() // Main function
{
int a,b,i,t,j,x,n,k=0,flag=0,prime[100]; // Variable declaration
char m[20],pp[20];
float p[20],c[20]; // Variable for plain text and cipher text
double e,d; // Variable for encryption and decryption key
clrscr();
for(i=0;i<50;i++) // Checks whether the given number is prime number
{
        flag=0;
        for(j=2;j<i/2;j++)
        if(i%j==0)
        {
            flag=0;
            break;
        }
        if(flag==0)
        prime[k++]=i;
        }
a=prime[k-1];  //  First prime number
b=prime[k-2]; //  Second prime number
n=a*b; // Multiplying two prime number to generate n
t=(a-1)*(b-1); // Malculating totient value from two prime number
e=(double)prime[2];  // Number e as a derived number which should be greater than 1 and
less than (p-1) and (q-1)
d=1/(float)e;  // Private Key d is calculated from the numbers p, q and e
printf("\n key of encryption is %lf",d); // %lf for double
printf("\n enter the text:");
scanf("%s",&m);  // Message to encrypt using RSA algorithm
x=strlen(m); // Length of string is measured
printf("\n \t source\t\t\t\tdestination");
printf("\nchar  numeric  \tcipher  \tnumeric  \tchar \n");
for(i=0;i<x;i++) // Encryption continued till total length of  the string
{
printf("%c",m[i]); // Plain text is printed
printf("\t%d",m[i]-97);
c[i]=pow(m[i]-97,e);
c[i]=fmod(c[i],n); // Cipher text  is created using equation   Me  mod n
printf("\t%f",c[i]); // Cipher text is created and printed
p[i]=pow(c[i],d);
p[i]=fmod(p[i],n); // Plain text is created using equation P= Cd mod n
printf("\t%f",p[i]);
pp[i]=p[i]+97;
printf("\t%c\n",pp[i]);
}
getch();
}
```

OUTPUT :

```
Enter 2 Prime Numbers: 5 7

F(n)=24

Enter e: 11

Public key: {11, 35}

Private key: {11, 35}

Enter the plain key: 13

Encrypted key: 27

Enter the Cipher key: 17

Decrypted key: 33
```

Ex No : 05 USER AUTHENTICATION SECURITY

AIM: To implement User Authentication Security.

User Authentication Security:

Authentication technology provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server. Users are usually identified with a user ID, and authentication is accomplished when the user provides a credential, for example a password, that matches with that user ID.

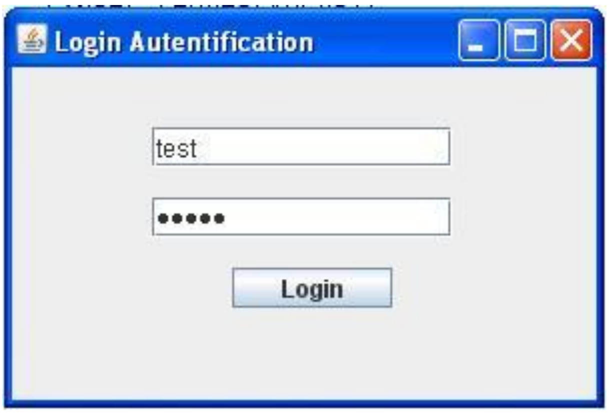
PROGRAM:

```
import java.awt.event.*;  // Header files
import java.awt.*;
import javax.swing.*;
public class Log extends JFrame
// Class declaration extends with JFrame to create GUI
{
public static void main(String[] args) // Main Method
 {
Log frameTabel = new Log(); //
}
JButton blogin = new JButton("Login"); //  To create Login Window
JPanel panel = new JPanel();  //
JTextField txuser = new JTextField(15);
JPasswordField pass = new JPasswordField(15);
Log()
{
super("Login Autentification"); // to print header information on panel
setSize(300,200);
setLocation(500,280);
panel.setLayout (null);
txuser.setBounds(70,30,150,20);
//The setBounds() method is used to set the position and size
pass.setBounds(70,65,150,20);
blogin.setBounds(110,100,80,20);
panel.add(blogin);
panel.add(txuser);
panel.add(pass);
getContentPane().add(panel);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
actionLogin();
}


public void actionLogin() // Method to perform authentication
{
blogin.addActionListener(new ActionListener())
// Actionlistener is applied on button
 {
public void actionPerformed(ActionEvent ae)
// The actionPerformed() method is invoked automatically whenever you click on the
registered component.
 {
String puname = txuser.getText();
String ppaswd = pass.getText();
if(puname.equals("test") && ppaswd.equals("12345"))
// Checks whether the given user name and password matches or not
 {
newframe regFace =new newframe();
regFace.setVisible(true);
dispose();
}
else
// To display message for wrong password
 {

JOptionPane.showMessageDialog(null,"Wrong Password / Username");
txuser.setText("");
pass.setText("");
txuser.requestFocus();
}
}
}
);
}
}
```

OUTPUT:



PROGRAM 6 : DYNAMIC PAGE GENERATION

AIM :

To write a Java Program to create Dynamic Page Generation using Java Swings

JAVA SWING:

Swing is a set of program component s for **Java** programmers that provide the ability to create graphical user interface (GUI) components, such as buttons and scroll bars, that are independent of the windowing system for specific operating system .

Swing components are used with the **Java Foundation Classes** (JFC)

PROCEDURE TO RUN A PROGRAM

Step 1: Open the Command Prompt and Set the path **C:\JDK1.5\BIN\.**

Step 2: File > **Notepad** and then write your Java program

Step 3: Save the program using the extension should be **".java**".

Step 4: Compile the program using **Javac Filename.java**

Step 5: Run the program using **Java Filename.**

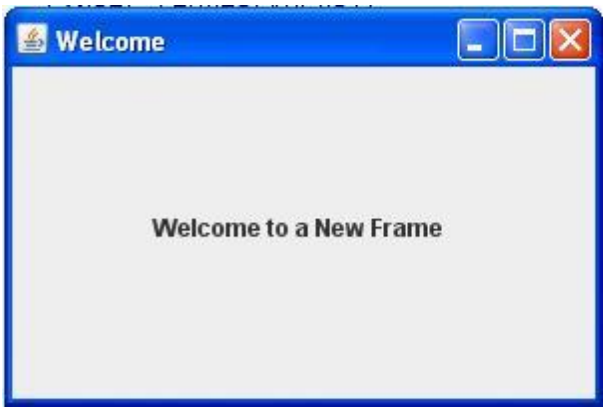
<u>PROGRAM:</u>

```
import  javax.swing.*; //Include Java Swing and AWT Packages
import  java.awt.*;
import  java.awt.event.*;
public class newframe extends JFrame //Create a classname new Frame
 {
public static void main(String[] args)
{
// Create a object for the java Frame
newframe frame Label = new newframe();

}
// Create a object for the java Label
JLabel welcome = new JLabel("Welcome to a New Frame");
// Create a object for the java JPanel

JPanel  panel = new JPanel();
newframe() //Constructor Name
{
super("Welcome");
setSize(300,200); //Set the size for the window
setLocation(500,280);
panel.setLayout (null);
welcome.setBounds(70,50,150,60);
panel.add(welcome);
getContentPane().add(panel); //Add the panel
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setVisible(true);
}
}
```

OUTPUT:



Welcome to a New Frame