PROGRAM 1 : Diffie Helman using JavaScript and HTML

Code:

dh_client.html

```
<html>

<body>
   <script>
      var headers = new Headers();

      //setting the headers for the server communication because of CORS
      headers.append('Content-Type', 'application/json');
      headers.append('Accept', 'application/json');
      headers.append('Origin', 'http://localhost:5000');

      var flag = 0;
      var key = 0;

      function display(message) {
         document.getElementById("message").innerHTML = message;
         document.getElementById("ka").innerHTML = "";
         document.getElementById("kb").innerHTML = "";

      }

      //converting the binary format to ASCII
      function toAscii(input) {
         var result = "";
         var arr = input.match(/.{1,8}/g);
         for (var i = 0; i < arr.length; i++) {
            result += String.fromCharCode(parseInt(arr[i], 2).toString(10));
         }
         return result;
      }

      //encodes of decodes based on the pressed button
      function coder(mode) {
         display(mode)
```

```
    fetch(`http://localhost:5000/${mode}/${key}`, {
        mode: 'cors',
        credentials: 'include',
        method: 'GET',
        headers: headers
    }).then(function (response) {
        return response.json();
    }).then(function (jsonResponse) {
        console.log(mode)
        if (mode == "Encode") {
            //convert the binary stuff to ASCII
            display(toAscii(jsonResponse.message));
        }
        else {
            display(jsonResponse.message);
        }
    });
}

//generates the buttons after the key has been generated
function createButton(message) {
    var button = document.createElement("BUTTON");
    var buttonText = document.createTextNode(message);

    button.setAttribute('onclick', 'coder("' + message + '")');
    button.appendChild(buttonText);
    document.body.appendChild(button);
}

//produces Ka and Kb based on the given public keys
function myFunction() {
    key = document.getElementById("key").value;
    fetch(`http://localhost:5000/getParam/${key}`, {
        mode: 'cors',
        credentials: 'include',
        method: 'GET',
        headers: headers
    }).then(function (response) {
        return response.json();
    }).then(function (jsonResponse) {
        display(jsonResponse.message)
        if (jsonResponse.message == "ka and kb are:") {
            if (flag == 0) {
                //if no button is created then create the buttons
```

```
                createButton("Encode");
                createButton("Decode");
                flag = 1;
            } else if (jsonResponse.message == "'message': 'deleted session'") {
                flag = 0;
            }
            document.getElementById("ka").innerHTML = jsonResponse.ka;
            document.getElementById("kb").innerHTML = jsonResponse.kb;
            key = jsonResponse.kb;
        }
    });
  }
</script>


<p>Enter the key:</p>
<p><input id="key" type="text" name="key" /></p>
<button onclick="myFunction()">Click me</button>

<p id="message"></p>
<p id="ka"></p>
<p id="kb"></p>
<p id="coder"></p>
</body>

</html>
```

dh_server.py

```
#importing flask server stuff
from flask import Flask, redirect, url_for, request, session, jsonify
from flask_cors import CORS, cross_origin

#this is for calling other processes (the DES C-code)
from subprocess import call
import random

app = Flask(__name__)
app.secret_key = 'hello there'
CORS(app, support_credentials=True)
```

```
#setting the variables
P = 23
G = 9
a = -1
b = -1
cache = 0

#killing the session should the users want to create a new key based on different inputs
def kill_session():
    global cache, a, b
    cache = 0
    a = -1
    b = -1
    return jsonify({'message': 'deleted session'})


#diffie helman to genarate the key
def keyGen(a, b):
    key = 0
    x = int(pow(G, a, P))
    y = int(pow(G, b, P))
    ka = int(pow(y, a, P))
    kb = int(pow(x, b, P))
    return ka, kb


@app.route('/getParam/<body>', methods=['POST', 'GET'])
@cross_origin(supports_credentials=True)
def success(body):
    global cache, a, b

    if body == "clear":
        return kill_session()

    #keeps track of the users as it needs 2 to form the key
    else:
        if cache == 0:
            a = int(body)
            cache = 1
        elif cache == 1:
            b = int(body)
            cache = 2
```

```
        #if both public keys are present then scale the key up and set size to 64 for the DES
        if a > -1 and b > -1 and cache != 0:
            ka, kb = keyGen(a, b)
            offset = random.randint(20,50)
            scalar = random.randint(20, 50)
            ka_pow_bin = bin(pow(offset+18, scalar))[2:66]
            ka_pow = int(ka_pow_bin,2)
            res = jsonify({'ka': ka_pow, 'kb': ka_pow_bin, 'message': 'ka and kb are:'})
        else:
            res = jsonify({'message': "waiting for other user"})

        return res


@app.route('/<mode>/<key>', methods=['POST', 'GET'])
@cross_origin(supports_credentials=True)
def coder(mode, key):
    #write the key that was produced into the file
    key_file = open("../dh/DES/inputs/key.txt", "w+")
    key_file.write(key)
    key_file.close()

    res = ""

    #based on the operation call the respective file and send to webpage
    if mode == "Decode":
        call(["./des", "d"])
        res = open("DES/output/result.txt", "r").read()
    else:
        call(["./des", "e"])
        res = open("DES/output/cipher.txt", "r").read()

    return jsonify({"message":res})


if __name__ == '__main__':
    app.run(debug=True)
```

des.c (same as from week4)

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#include <time.h>

int IP[] =
   {
      58, 50, 42, 34, 26, 18, 10, 2,
      60, 52, 44, 36, 28, 20, 12, 4,
      62, 54, 46, 38, 30, 22, 14, 6,
      64, 56, 48, 40, 32, 24, 16, 8,
      57, 49, 41, 33, 25, 17, 9, 1,
      59, 51, 43, 35, 27, 19, 11, 3,
      61, 53, 45, 37, 29, 21, 13, 5,
      63, 55, 47, 39, 31, 23, 15, 7};

int E[] =
   {
      32, 1, 2, 3, 4, 5,
      4, 5, 6, 7, 8, 9,
      8, 9, 10, 11, 12, 13,
      12, 13, 14, 15, 16, 17,
      16, 17, 18, 19, 20, 21,
      20, 21, 22, 23, 24, 25,
      24, 25, 26, 27, 28, 29,
      28, 29, 30, 31, 32, 1};

int P[] =
   {
      16, 7, 20, 21,
      29, 12, 28, 17,
      1, 15, 23, 26,
      5, 18, 31, 10,
      2, 8, 24, 14,
      32, 27, 3, 9,
      19, 13, 30, 6,
      22, 11, 4, 25};

int FP[] =
   {
      40, 8, 48, 16, 56, 24, 64, 32,
      39, 7, 47, 15, 55, 23, 63, 31,
      38, 6, 46, 14, 54, 22, 62, 30,
      37, 5, 45, 13, 53, 21, 61, 29,
```

```
        36, 4, 44, 12, 52, 20, 60, 28,
        35, 3, 43, 11, 51, 19, 59, 27,
        34, 2, 42, 10, 50, 18, 58, 26,
        33, 1, 41, 9, 49, 17, 57, 25};

int S1[4][16] =
    {
        14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
        0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
        4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
        15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13};

int S2[4][16] =
    {
        15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
        3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
        0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
        13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9};

int S3[4][16] =
    {
        10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
        13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
        13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
        1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12};

int S4[4][16] =
    {
        7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
        13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
        10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
        3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14};

int S5[4][16] =
    {
        2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
        14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
        4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
        11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3};

int S6[4][16] =
    {
        12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
        10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
```

```
        9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
        4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13};

int S7[4][16] =
   {
        4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
        13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
        1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
        6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12};

int S8[4][16] =
   {
        13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
        1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
        7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
        2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11};

int PC1[] =
   {
        57, 49, 41, 33, 25, 17, 9,
        1, 58, 50, 42, 34, 26, 18,
        10, 2, 59, 51, 43, 35, 27,
        19, 11, 3, 60, 52, 44, 36,
        63, 55, 47, 39, 31, 23, 15,
        7, 62, 54, 46, 38, 30, 22,
        14, 6, 61, 53, 45, 37, 29,
        21, 13, 5, 28, 20, 12, 4};

int PC2[] =
   {
        14, 17, 11, 24, 1, 5,
        3, 28, 15, 6, 21, 10,
        23, 19, 12, 4, 26, 8,
        16, 7, 27, 20, 13, 2,
        41, 52, 31, 37, 47, 55,
        30, 40, 51, 45, 33, 48,
        44, 49, 39, 56, 34, 53,
        46, 42, 50, 36, 29, 32};

int SHIFTS[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

FILE *out;
int LEFT[17][32], RIGHT[17][32];
int IPtext[64];
```

```
int EXPtext[48];
int XORtext[48];
int X[8][6];
int X2[32];
int R[32];
int key56bit[56];
int key48bit[17][48];
int CIPHER[64];
int ENCRYPTED[64];

void expansion_function(int pos, int text)
{
    for (int i = 0; i < 48; i++)
        if (E[i] == pos + 1)
            EXPtext[i] = text;
}

int initialPermutation(int pos, int text)
{
    int i;
    for (i = 0; i < 64; i++)
        if (IP[i] == pos + 1)
            break;
    IPtext[i] = text;
}

int F1(int i)
{
    int r, c, b[6];
    for (int j = 0; j < 6; j++)
        b[j] = X[i][j];

    r = b[0] * 2 + b[5];
    c = 8 * b[1] + 4 * b[2] + 2 * b[3] + b[4];
    if (i == 0)
        return S1[r][c];
    else if (i == 1)
        return S2[r][c];
    else if (i == 2)
        return S3[r][c];
    else if (i == 3)
        return S4[r][c];
    else if (i == 4)
        return S5[r][c];
```

```
    else if (i == 5)
        return S6[r][c];
    else if (i == 6)
        return S7[r][c];
    else if (i == 7)
        return S8[r][c];
}

int XOR(int a, int b)
{
    return (a ^ b);
}

int ToBits(int value)
{
    int k, j, m;
    static int i;
    if (i % 32 == 0)
        i = 0;
    for (j = 3; j >= 0; j--)
    {
        m = 1 << j;
        k = value & m;
        if (k == 0)
            X2[3 - j + i] = '0' - 48;
        else
            X2[3 - j + i] = '1' - 48;
    }
    i = i + 4;
}

int SBox(int XORtext[])
{
    int k = 0;
    for (int i = 0; i < 8; i++)
        for (int j = 0; j < 6; j++)
            X[i][j] = XORtext[k++];

    int value;
    for (int i = 0; i < 8; i++)
    {
        value = F1(i);
        ToBits(value);
    }
```

```
}

int PBox(int pos, int text)
{
    int i;
    for (i = 0; i < 32; i++)
        if (P[i] == pos + 1)
            break;
    R[i] = text;
}

void cipher(int Round, int mode)
{
    for (int i = 0; i < 32; i++)
        expansion_function(i, RIGHT[Round - 1][i]);

    for (int i = 0; i < 48; i++)
    {
        if (mode == 0)
            XORtext[i] = XOR(EXPtext[i], key48bit[Round][i]);
        else
            XORtext[i] = XOR(EXPtext[i], key48bit[17 - Round][i]);
    }

    SBox(XORtext);

    for (int i = 0; i < 32; i++)
        PBox(i, X2[i]);
    for (int i = 0; i < 32; i++)
        RIGHT[Round][i] = XOR(LEFT[Round - 1][i], R[i]);
}

void finalPermutation(int pos, int text)
{
    int i;
    for (i = 0; i < 64; i++)
        if (FP[i] == pos + 1)
            break;
    ENCRYPTED[i] = text;
}

void convertToBinary(int n)
{
    int k, m;
```

```
    for (int i = 7; i >= 0; i--)
    {
        m = 1 << i;
        k = n & m;
        if (k == 0)
            fprintf(out, "0");
        else
            fprintf(out, "1");
    }
}

int convertCharToBit(long int n)
{
    FILE *inp = fopen("../exe/DES/inputs/input.txt", "rb");
    out = fopen("../exe/DES/output/bits.txt", "wb+");
    char ch;
    int i = n * 8;
    while (i)
    {
        ch = fgetc(inp);
        if (ch == -1)
            break;
        i--;
        convertToBinary(ch);
    }
    fclose(out);
    fclose(inp);
}

void Encryption(long int plain[])
{
    out = fopen("../exe/DES/output/cipher.txt", "ab+");
    for (int i = 0; i < 64; i++)
        initialPermutation(i, plain[i]);

    for (int i = 0; i < 32; i++)
        LEFT[0][i] = IPtext[i];
    for (int i = 32; i < 64; i++)
        RIGHT[0][i - 32] = IPtext[i];

    for (int k = 1; k < 17; k++)
    {
        cipher(k, 0);
```

```
        for (int i = 0; i < 32; i++)
            LEFT[k][i] = RIGHT[k - 1][i];
    }

    for (int i = 0; i < 64; i++)
    {
        if (i < 32)
            CIPHER[i] = RIGHT[16][i];
        else
            CIPHER[i] = LEFT[16][i - 32];
        finalPermutation(i, CIPHER[i]);
    }

    for (int i = 0; i < 64; i++)
        fprintf(out, "%d", ENCRYPTED[i]);
    fclose(out);
}

void Decryption(long int plain[])
{
    out = fopen("../exe/DES/output/decrypted.txt", "ab+");
    for (int i = 0; i < 64; i++)
        initialPermutation(i, plain[i]);

    for (int i = 0; i < 32; i++)
        LEFT[0][i] = IPtext[i];

    for (int i = 32; i < 64; i++)
        RIGHT[0][i - 32] = IPtext[i];

    for (int k = 1; k < 17; k++)
    {
        cipher(k, 1);

        for (int i = 0; i < 32; i++)
            LEFT[k][i] = RIGHT[k - 1][i];
    }
    for (int i = 0; i < 64; i++)
    {
        if (i < 32)
            CIPHER[i] = RIGHT[16][i];
        else
            CIPHER[i] = LEFT[16][i - 32];
        finalPermutation(i, CIPHER[i]);
```

```
    }
    for (int i = 0; i < 64; i++)
        fprintf(out, "%d", ENCRYPTED[i]);

    fclose(out);
}

void convertToBits(int ch[])
{
    int value = 0;
    for (int i = 7; i >= 0; i--)
        value += (int)pow(2, i) * ch[7 - i];
    fprintf(out, "%c", value);
}

int bittochar()
{
    out = fopen("../exe/DES/output/result.txt", "ab+");
    for (int i = 0; i < 64; i = i + 8)
        convertToBits(&ENCRYPTED[i]);
    fclose(out);
}

void key56to48(int round, int pos, int text)
{
    int i;
    for (i = 0; i < 56; i++)
        if (PC2[i] == pos + 1)
            break;
    key48bit[round][i] = text;
}

int key64to56(int pos, int text)
{
    int i;
    for (i = 0; i < 56; i++)
        if (PC1[i] == pos + 1)
            break;
    key56bit[i] = text;
}

void key64to48(unsigned int key[])
{
    int k, backup[17][2];
```

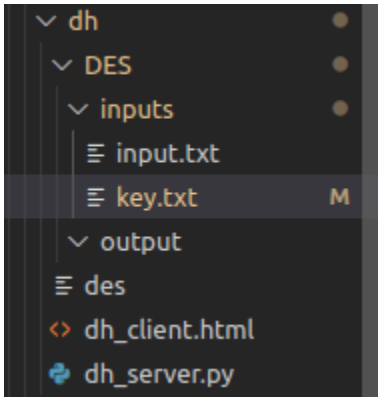
```
int CD[17][56];
int C[17][28], D[17][28];

for (int i = 0; i < 64; i++)
    key64to56(i, key[i]);

for (int i = 0; i < 56; i++)
    if (i < 28)
        C[0][i] = key56bit[i];
    else
        D[0][i - 28] = key56bit[i];

for (int x = 1; x < 17; x++)
{
    int shift = SHIFTS[x - 1];

    for (int i = 0; i < shift; i++)
        backup[x - 1][i] = C[x - 1][i];
    for (int i = 0; i < (28 - shift); i++)
        C[x][i] = C[x - 1][i + shift];
    k = 0;
    for (int i = 28 - shift; i < 28; i++)
        C[x][i] = backup[x - 1][k++];

    for (int i = 0; i < shift; i++)
        backup[x - 1][i] = D[x - 1][i];
    for (int i = 0; i < (28 - shift); i++)
        D[x][i] = D[x - 1][i + shift];
    k = 0;
    for (int i = 28 - shift; i < 28; i++)
        D[x][i] = backup[x - 1][k++];
}

for (int j = 0; j < 17; j++)
{
    for (int i = 0; i < 28; i++)
        CD[j][i] = C[j][i];
    for (int i = 28; i < 56; i++)
        CD[j][i] = D[j][i - 28];
}

for (int j = 1; j < 17; j++)
    for (int i = 0; i < 56; i++)
        key56to48(j, i, CD[j][i]);
```

```
}

void decrypt(long int n)
{
   FILE *in = fopen("../exe/DES/output/cipher.txt", "rb");
   long int plain[n * 64];
   int i = -1;
   char ch;

   while (!feof(in))
   {
      ch = getc(in);
      plain[++i] = ch - 48;
   }

   for (int i = 0; i < n; i++)
   {
      Decryption(plain + i * 64);
      bittochar();
   }
   fclose(in);
}

void encrypt(long int n)
{
   FILE *in = fopen("../exe/DES/output/bits.txt", "rb");

   long int plain[n * 64];
   int i = -1;
   char ch;

   while (!feof(in))
   {
      ch = getc(in);
      plain[++i] = ch - 48;
   }

   for (int i = 0; i < n; i++)
      Encryption(plain + 64 * i);

   fclose(in);
}

void create16Keys()
```

```
{
    FILE *pt = fopen("../exe/DES/inputs/key.txt", "rb");
    unsigned int key[64];
    int i = 0, ch;

    while (!feof(pt))
    {
        ch = getc(pt);
        key[i++] = ch - 48;
    }

    key64to48(key);
    fclose(pt);
}

long int findFileSize()
{
    FILE *inp = fopen("../exe/DES/inputs/input.txt", "rb");
    long int size;
    if (fseek(inp, 0L, SEEK_END))
        perror("fseek() failed");
    else // size will contain no. of chars in input file.
        size = ftell(inp);
    fclose(inp);

    return size;
}

void cleanFiles(){
    out = fopen("../exe/DES/output/result.txt", "wb+");
    fclose(out);
    out = fopen("../exe/DES/output/decrypted.txt", "wb+");
    fclose(out);
    out = fopen("../exe/DES/output/cipher.txt", "wb+");
    fclose(out);
}

int main(int argc, char **argv)
{
    create16Keys();

    long int n = findFileSize() / 8;

    convertCharToBit(n);
```

```
    if(*argv[1] == 'e'){
        cleanFiles();
        encrypt(n);
    } else{
        decrypt(n);
    }
    return(0);
}
```

Output:




```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week9/dh/DES/inputs$ cat key.txt
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week9/dh/DES/inputs$ cat input.txt && echo ""
hello how are you doing?
```

Enter the key:

Click me

Enter the key:

5

Click me

waiting for other user

Enter the key:

9

Click me

ka and kb are:

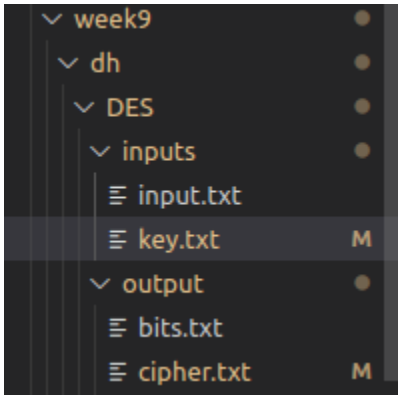
12252192985362455000

1010101000001000100010001001101011110100000011111101010101100101

Encode Decode

(kb here stands for kbinary and is just the representation of ka in binary)

```
shankar@shankar-ThinkPad-L450:~/Documents/AU/sem6/security/lab/week9/dh/DES/inputs$ cat key.txt && echo ""
1010101000001000100010001001101011110100000011111101010101100101
```



On clicking encode

Enter the key:

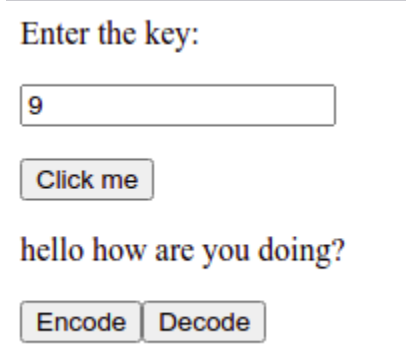
9

Click me

♫Òò♫¯)(ÙSrM¼$*±`r∝ ÈPÙ

Encode Decode

On clicking decode



PROGRAM 2 : SHA-1

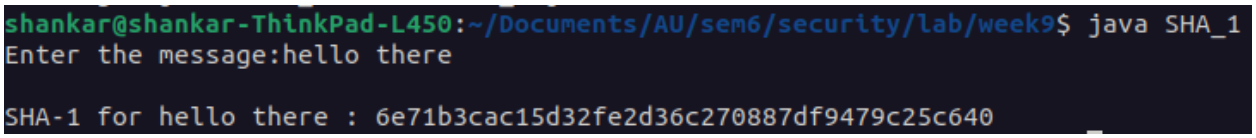
Code:

SHA_1.java

```
import java.io.*;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class SHA_1 {
    public static String messageDigest(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");

            byte[] messageDigest = md.digest(input.getBytes());

            BigInteger num = new BigInteger(1, messageDigest);

            String hashtext = num.toString(16);

            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }

            return hashtext;
        }
```

```
        catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    public static void main(String args[]) throws NoSuchAlgorithmException, IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

        System.out.print("Enter the message:");
        String str = br.readLine();

        System.out.println("\nSHA-1 for " + str + " : " + messageDigest(str));
    }
}
```

Output:



PROGRAM 3 : Login and Registration using Hashing

Code:

Login.java

```
package com.mycompany.marketplace;

import com.mycompany.DBStuff.DBStuff;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
```

```
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

@WebServlet("/login")
public class Login extends HttpServlet {

        public static String messageDigest(String input) {
        try {
            MessageDigest md = MessageDigest.getInstance("SHA-1");

            byte[] messageDigest = md.digest(input.getBytes());

            BigInteger num = new BigInteger(1, messageDigest);

            String hashtext = num.toString(16);

            while (hashtext.length() < 32) {
                hashtext = "0" + hashtext;
            }

            return hashtext;
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);
        }
    }

    //get the parameters and check if the user's account is there
    public void login(HttpServletRequest request, HttpServletResponse response, PrintWriter out,
DBStuff db) throws SQLException {
        String usr = request.getParameter("usr");
        String pwd = request.getParameter("pwd");

        boolean loginStatus = db.checkLogin(usr, messageDigest(pwd));
        out.println("Login ? " + loginStatus);

        if (loginStatus) {
            setSessionLogin(request, usr);
        }
```

```
    }

    public void setSessionLogin(HttpServletRequest request, String usr) throws SQLException {
        HttpSession session = request.getSession();
        session.setAttribute("usr", usr);

    }

    //sign the user up by calling the database and passing the pwd as a parameter to
messageDigest
    public void signup(HttpServletRequest request, HttpServletResponse response, PrintWriter
out, DBStuff db) throws SQLException {

        String card = request.getParameter("card");
        int cardLen = card.length();

        if (cardLen > 0) {
            String usr = request.getParameter("usr");
            String email = request.getParameter("email");
            String pwd = request.getParameter("pwd");

            boolean signUpStatus = db.registerUser(usr, messageDigest(pwd), email, card);
            out.println("Sign-Up ? " + signUpStatus);
            out.println("Sign-Up ? " + messageDigest(pwd));



        } else {
            getSignUpDets(out);
        }
    }

    public void getSignUpDets(PrintWriter out) {
        out.println("<center><form method = 'POST' action = 'login'>");
        out.println("Name:   <input type = 'text'     name = 'usr'   required> </br>"
                + "Password: <input type = 'password' name = 'pwd'   required> </br>"
                + "Email:    <input type = 'email'    name = 'email' required> </br>"
                + "Card:     <input type = 'card'     name = 'card'  required> </br>");
        out.println("<input type = 'submit' value = 'sign up'> </form></center>");
    }

    public void loginForm(PrintWriter out, String returnPage) {
        out.println("<form action = 'login?return=" + returnPage + "' method = 'POST' >"
                + "<br> Name: <input type = 'text' name = 'usr'>"
                + " Password: <input type = 'password' name = 'pwd'>"
```

```
            + "<input type = 'submit' name = 'sign-handler' value = 'login'>"
            + "<input type = 'submit' name = 'sign-handler' value = 'sign-up'> </form><br>");

}

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException, SQLException {
    response.setContentType("text/html;charset=UTF-8");
    try ( PrintWriter out = response.getWriter()) {

        String signHandler = request.getParameter("sign-handler");
        String returnPage = request.getParameter("return");
        if ("login".equals(signHandler)) {
            login(request, response, out, new DBStuff());
            out.println("<meta http-equiv='Refresh' content=\"5; url='" + returnPage + "'\" />");

        } else if ("logout".equals(signHandler)) {
            setSessionLogin(request, null);
             out.println("<meta http-equiv='Refresh' content=\"0.5; url='" + returnPage + "'\" />");

        } else if ("signup".equals(signHandler)){
            getSignUpDets(out);
        } else{
             signup(request, response, out, new DBStuff());
            out.println("<meta http-equiv='Refresh' content=\"0.5; url='" + returnPage + "'\" />");

        }

    }
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    try {
        processRequest(request, response);
    } catch (SQLException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }
}

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

```
        response.setContentType("text/html;charset=UTF-8");

        try ( PrintWriter out = response.getWriter()) {
            loginForm(out, request.getParameter("return"));
        }
    }

    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

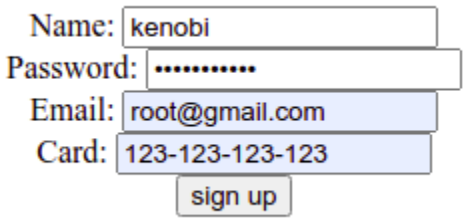
DBStuff.java

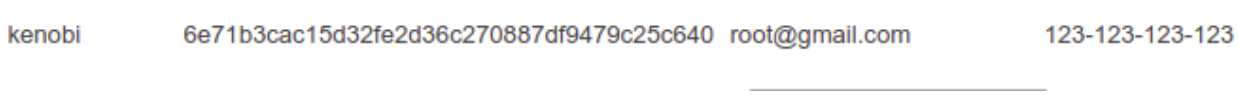
```
package com.mycompany.DBStuff;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

//basics database stuff here
public class DBStuff implements DBStuffRemote {

    Connection conn = null;



    @Override
    public Connection getConn(String table) throws ClassNotFoundException, SQLException {

        String url = "jdbc:mysql://localhost:3306/" + table;

        String username = "root";
        String password = "";

        try {
```

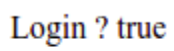
```
            Class.forName("com.mysql.jdbc.Driver");

            conn = DriverManager.getConnection(url, username, password);

        } catch (SQLException ex) {
            System.out.println("in exec");
            System.out.println(ex.getMessage());
        }

        return conn;
    }

    @Override
    public boolean registerUser(String usr, String pwd, String email, String card) throws
SQLException {
        boolean flag = false;
        String query = "INSERT INTO ACCOUNTS VALUES( '" + usr + "','" + pwd + "','" + email +
"','" + card + "')";

        try {
            conn = getConn("MARKETPLACE");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(DBStuff.class.getName()).log(Level.SEVERE, null, ex);
        }

        PreparedStatement pstmt = conn.prepareStatement(query);
        int rs = pstmt.executeUpdate();

        flag = rs > 0;

        conn.close();
        return flag;
    }

    public boolean checkLogin(String usr, String pwd) throws SQLException {
        boolean flag = false;
        String query = "SELECT * FROM  ACCOUNTS WHERE NAME = '" + usr + "' AND PWD =
'" + pwd + "'";

        try {
            conn = getConn("MARKETPLACE");
        } catch (ClassNotFoundException ex) {
            Logger.getLogger(DBStuff.class.getName()).log(Level.SEVERE, null, ex);
        }
```

```
        Statement selectStmt = conn.createStatement();
        ResultSet rs = selectStmt
                .executeQuery(query);

        while (rs.next()) {
            flag = true;
        }

        conn.close();
        return flag;
    }
}
```

Output:







Pwd used was hello there
Hashes to 6e71b3cac15d32fe2d36c270887df9479c25c640