# LAB ASSESSMENT 5

**Name:** Charan Lalchand Soneji

**Registration number:** 17BCE2196

**Faculty:** Prof Marimuthu K.

**Course Title:** Cryptography fundamentals

**Slot:** L51+ L52

**10. To implement the any one of the Hash technique**

**CODE:**

```
#include<stdio.h>
int main()
{
char ch, input[9], s[9];
int i, len=0;

printf("Enter your input bits:");
scanf("%s",input);

printf("Enter your key S (8 bits):");
scanf("%s",s);

printf("Your hash code is:");
for(i=0; i<8; i++)
{
ch=input[i]^s[i];
printf("%d",ch);
}
return 0;
}
```

**OUTPUT:**

```
Enter your input bits:101101101
Enter your key S (8 bits):00101111
Your hash code is:10011001
Process returned 0 (0x0)   execution time : 12.379 s
Press ENTER to continue.
```

11. Demonstration of authentication technique with graphical user user interface(GUI)

CODE:

```
from PyQt4 import QtCore, QtGui
import smtplib
import string
import socket
import random
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
from recovery_ui import *
from home_ui import *
import mysql.connector
import imageselect
from mysql.connector import *
from mysql.connector import Error,MySQLConnection
from image_retrieve import *
import json
try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)



class Ui_Login(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        self.setupUi(self)
        self.imgRetrieve = imageRetrieve()
        self.temp=0
        self.imagenum=0                 #counter for the images shown
        self.imgRetrieve.main()           # main() function of the imageRetrieve class retrieves all the images from the
database

    def setupUi(self, Login):
        Login.setObjectName(_fromUtf8("Login"))
        Login.resize(600, 400)
        Login.setFixedSize(600,400)
        Login.setWindowModality(QtCore.Qt.ApplicationModal)
        self.loginimage = ExtendedQLabel(Login)
        self.loginimage.setGeometry(QtCore.QRect(0, 0, 600, 400))
        self.loginimage.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
        self.loginimage.setObjectName(_fromUtf8("loginimage"))
        self.loginimage.setScaledContents(True)

        self.errorlabel = QtGui.QLabel(Login)
        self.errorlabel.setGeometry(QtCore.QRect(190, 215, 200, 50))
        palette = QtGui.QPalette()
```

```
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
        self.errorlabel.setPalette(palette)
        font = QtGui.QFont()
        font1 = QtGui.QFont()
        font.setPointSize(10)
        font1.setPointSize(10)
        font.setItalic(True)
        self.errorlabel.setFont(font)
        self.errorlabel.setText("")

        self.forgotButton=QtGui.QPushButton(Login)
        self.forgotButton.setGeometry(QtCore.QRect(300, 200, 100, 30))
        self.forgotButton.setText("Forgot Password")

        self.clickpoint = QtGui.QPushButton(Login)
        self.clickpoint.setGeometry(QtCore.QRect(330, 90, 75, 51))
        self.clickpoint.setObjectName(_fromUtf8("clickpoint"))
        self.okay = QtGui.QPushButton(Login)
        self.okay.setGeometry(QtCore.QRect(190, 200, 100, 30))
        self.okay.setObjectName(_fromUtf8("Okay"))
        self.okay.setText("Login")
        self.lineEdit = QtGui.QLineEdit(Login)
        self.lineEdit.setGeometry(QtCore.QRect(190, 160, 210, 30))
        self.lineEdit.setObjectName(_fromUtf8("lineEdit"))
        self.lineEdit.setFont(font1)
        self.retranslateUi(Login)
        self.okay.clicked['bool'].connect(self.GetUsername)
        self.clickpoint.setStyleSheet(_fromUtf8("\n"
                                      "background-color: rgb(0, 0, 0,0);\n"))
        self.clickpoint.setDisabled(True)
        self.clickpoint.clicked['bool'].connect(self.gencorrectImage)
        self.forgotButton.clicked['bool'].connect(self.emailfunc)
        self.clickpoint.setFlat(True)
        self.connect(self.loginimage,SIGNAL("clicked()"),self.genrandomImage)
        QtCore.QMetaObject.connectSlotsByName(Login)


    def retranslateUi(self, Login):
        Login.setWindowTitle(_translate("Login", "Login", None))
        self.lineEdit.setPlaceholderText(_translate("Login", "username", None))


#sending mail for recovery
    def emailfunc(self):
        self.uname=str(self.lineEdit.text())
        if not self.uname:
            self.errorlabel.setText("***Enter correct username***")

        else:
                try:
                    self.code=''
                    for i in range(8):
                        send=choice(string.ascii_letters)
                        self.code=self.code+send
```

```
                self.db=mysql.connector.connect(host="localhost",
database="python_mysql",user="root",password="2864")
                self.cursor=self.db.cursor()
                self.temp=(self.code,self.uname)
                self.cursor.execute("UPDATE registration set recovery_txt=%s where Username=%s",self.temp)
                self.db.commit()
                self.cursor.execute("select * from registration where username=%s",(self.uname,))
                self.rows=self.cursor.fetchone()
                self.send_mail(self.rows[4])
                QMessageBox.information(self, 'Check your Email!','An email has been sent to your
account with the recovery text.')
                self.close()
                self.recover = Ui_Recovery(self.uname,self.code)                    #recovery code and username are passed as
arguments
                self.recover.show()
            except Error as e:
                errorstr = str(e)
                QtGui.QMessageBox.warning(self, 'Sorry!', errorstr)
            except TypeError:
                self.errorlabel.setText("***Username not found***")
            except smtplib.SMTPAuthenticationError as e:
                strerr = str(e)
                QtGui.QMessageBox.warning(self, 'Sorry!', strerr +"\n\n\nSystem could not login. Try again later!")
            except TimeoutError as e:
                strerr = str(e)
                QtGui.QMessageBox.warning(self, 'Sorry!', strerr +"\n\n\nSystem could not login. Try again later!")
            #except:
            #    QtGui.QMessageBox.warning(self, 'Sorry!', "Something went wrong. Try again later!")

            finally:
                self.cursor.close()
                self.db.close()


#send_mail function sends the recovery text from iamnotsparsha@gmail.com to the
    user Email def send_mail(self,userEmail):
        #gmail connection
        try:
            mail=smtplib.SMTP('smtp.gmail.com:587')
            mail.ehlo()
            mail.starttls()
            mail.login('iamnotsparsha@gmail.com', 'fantasy.101')
            content = "Please use this code to log into your account: "
            content = 'Subject: %s\n\n%s' % ("Password Reset", content)
            content=content+ self.code
            mail.sendmail('iamnotsparsha@gmail.com',userEmail,content)
            mail.close()

        except UnboundLocalError as e:
            strerr = str(e)
            QtGui.QMessageBox.warning(self, 'Sorry!', strerr)

        finally:
          pass


    def GetUsername(self):
        #******************first the username is fetched from the Database*************************
        self.uname=str(self.lineEdit.text())
        if not self.uname:
            self.errorlabel.setText("***Enter a username***")
        else:
            try:
```

```
            self.db=mysql.connector.connect(host="localhost", database="python_mysql",user="root",password="2864")
            self.cursor=self.db.cursor()
            self.cursor.execute("select * from registration where username=%s",(self.uname,))
            rows=self.cursor.fetchone()
            if not rows:
               self.errorlabel.setText("***Invalid Username***")
            else:
               '''*************if the username is matched, user details like number of images(inum),
                        tolerance value(tvalue),list of images(imageList) and clickpoints(cpx,cpy)
                        are fetched from the database************************'''
               self.inum=rows[5]
               self.tvalue=rows[6]
               self.imageList=[]                        #list of user images
               self.cpx=[]                          #list of x-coordinates for clickpoints
               self.cpy=[]                          #list of Y-coordinates for clickpoints

               for i in range (self.inum):
                  self.new=json.loads(rows[i+7])
                  self.imageList.append(self.new[0])
                  self.cpx.append(self.new[1])
                  self.cpy.append(self.new[2])
               self.clickpoint.setDisabled(False)
               '''************after username is matched, clickpoint is enabled,
                        correct image is shown and remaining entities are
               closed***********''' self.gencorrectImage()
               self.lineEdit.close()
               self.okay.close()
               self.forgotButton.close()
               self.errorlabel.close()

         except Error as e:
            strerr = str(e)
            QtGui.QMessageBox.warning(self, 'Try again later!', strerr)
         finally:
            self.cursor.close()
            self.db.close()

#if the user clicks on correct series of clickpoints, correct series of images are
   generated def gencorrectImage(self):
      if self.imagenum <self.inum:
         self.loginimage.setPixmap(QtGui.QPixmap(str(self.imageList[self.imagenum])+".png"))

self.clickpoint.setGeometry(QtCore.QRect(self.cpx[self.imagenum],self.cpy[self.imagenum],self.tvalue,self.tvalue))
         self.imagenum = self.imagenum + 1 #increasing the counter else:

         self.imgRetrieve.remove_images()
         self.clickpoint.close()
         self.close()
         self.openHome(self.cpx,self.cpy)
      #self.temp=self.temp+1                    #don't remember what purpose this line of statement served


#f the user clicks on incorrect clickpoints, random series of images are
   displayed def genrandomImage(self):
      self.clickpoint.close()
      self.rand_num=randint(1,12)
      if self.imagenum <self.inum:
         self.loginimage.setPixmap(QtGui.QPixmap(str(self.rand_num)+".png"))
         self.imagenum = self.imagenum + 1 #increasing the counter else:

         self.close()
         QMessageBox.warning(self, 'Sorry!','Incorrect Password. Please try again!')
```

```
#if the user gets logged in then a new window, 'home' is
   shown def openHome(self,cpx,cpy):
      self.homeWindow = Ui_Home(cpx,cpy)
      self.homeWindow.show()



#class Definition for making the label clickable
class ExtendedQLabel(QLabel):

   def __init(self, parent):
      QLabel.__init__(self, parent)

 def mouseReleaseEvent(self, ev):
     self.emit(SIGNAL('clicked()'))

from main_window_ui import *
from registration_ui import *
import sys
import os

if __name__ == '__main__':
   app = QtGui.QApplication(sys.argv)
   mainWindow = Ui_MainWindow()
   mainWindow.show()
   sys.exit(app.exec())
```

## 12. Demonstration of authentication technique with graphical user user interface(GUI)

CODE:

```
from PyQt4 import QtCore, QtGui
import smtplib
import string
import socket
```

```
import random
from PyQt4.QtGui import *
from PyQt4.QtCore import *
import sys
from recovery_ui import *
from home_ui import *
import mysql.connector
import imageselect
from mysql.connector import *
from mysql.connector import Error,MySQLConnection
from image_retrieve import *
import json
try:
    _fromUtf8 = QtCore.QString.fromUtf8
except AttributeError:
    def _fromUtf8(s):
        return s

try:
    _encoding = QtGui.QApplication.UnicodeUTF8
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig, _encoding)
except AttributeError:
    def _translate(context, text, disambig):
        return QtGui.QApplication.translate(context, text, disambig)



class Ui_Login(QMainWindow):
    def __init__(self):
        QMainWindow.__init__(self)
        self.setupUi(self)
        self.imgRetrieve = imageRetrieve()
        self.temp=0
        self.imagenum=0                    #counter for the images shown
        self.imgRetrieve.main()              # main() function of the imageRetrieve class retrieves all the images from the
database

    def setupUi(self, Login):
        Login.setObjectName(_fromUtf8("Login"))
        Login.resize(600, 400)
        Login.setFixedSize(600,400)
        Login.setWindowModality(QtCore.Qt.ApplicationModal)
        self.loginimage = ExtendedQLabel(Login)
        self.loginimage.setGeometry(QtCore.QRect(0, 0, 600, 400))
        self.loginimage.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignTop)
        self.loginimage.setObjectName(_fromUtf8("loginimage"))
        self.loginimage.setScaledContents(True)

        self.errorlabel = QtGui.QLabel(Login)
        self.errorlabel.setGeometry(QtCore.QRect(190, 215, 200, 50))
        palette = QtGui.QPalette()
```

```
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.WindowText, brush)
        brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.WindowText, brush)
        self.errorlabel.setPalette(palette)
        font = QtGui.QFont()
        font1 = QtGui.QFont()
        font.setPointSize(10)
        font1.setPointSize(10)
        font.setItalic(True)
        self.errorlabel.setFont(font)
        self.errorlabel.setText("")

        self.forgotButton=QtGui.QPushButton(Login)
        self.forgotButton.setGeometry(QtCore.QRect(300, 200, 100, 30))
        self.forgotButton.setText("Forgot Password")

        self.clickpoint = QtGui.QPushButton(Login)
        self.clickpoint.setGeometry(QtCore.QRect(330, 90, 75, 51))
        self.clickpoint.setObjectName(_fromUtf8("clickpoint"))
        self.okay = QtGui.QPushButton(Login)
        self.okay.setGeometry(QtCore.QRect(190, 200, 100, 30))
        self.okay.setObjectName(_fromUtf8("Okay"))
        self.okay.setText("Login")
        self.lineEdit = QtGui.QLineEdit(Login)
        self.lineEdit.setGeometry(QtCore.QRect(190, 160, 210, 30))
        self.lineEdit.setObjectName(_fromUtf8("lineEdit"))
        self.lineEdit.setFont(font1)
        self.retranslateUi(Login)
        self.okay.clicked['bool'].connect(self.GetUsername)
        self.clickpoint.setStyleSheet(_fromUtf8("\n"
                                      "background-color: rgb(0, 0, 0,0);\n"))
        self.clickpoint.setDisabled(True)
        self.clickpoint.clicked['bool'].connect(self.gencorrectImage)
        self.forgotButton.clicked['bool'].connect(self.emailfunc)
        self.clickpoint.setFlat(True)
        self.connect(self.loginimage,SIGNAL("clicked()"),self.genrandomImage)
        QtCore.QMetaObject.connectSlotsByName(Login)


    def retranslateUi(self, Login):
        Login.setWindowTitle(_translate("Login", "Login", None))
        self.lineEdit.setPlaceholderText(_translate("Login", "username", None))


#sending mail for recovery
    def emailfunc(self):
        self.uname=str(self.lineEdit.text())
        if not self.uname:
            self.errorlabel.setText("***Enter correct username***")

        else:
                try:
                    self.code=''
                    for i in range(8):
                        send=choice(string.ascii_letters)
                        self.code=self.code+send
```

```
                self.db=mysql.connector.connect(host="localhost",
database="python_mysql",user="root",password="2864")
                self.cursor=self.db.cursor()
                self.temp=(self.code,self.uname)
                self.cursor.execute("UPDATE registration set recovery_txt=%s where Username=%s",self.temp)
                self.db.commit()
                self.cursor.execute("select * from registration where username=%s",(self.uname,))
                self.rows=self.cursor.fetchone()
                self.send_mail(self.rows[4])
                QMessageBox.information(self, 'Check your Email!','An email has been sent to your
account with the recovery text.')
                self.close()
                self.recover = Ui_Recovery(self.uname,self.code)                    #recovery code and username are passed as
arguments
                self.recover.show()
            except Error as e:
                errorstr = str(e)
                QtGui.QMessageBox.warning(self, 'Sorry!', errorstr)
            except TypeError:
                self.errorlabel.setText("***Username not found***")
            except smtplib.SMTPAuthenticationError as e:
                strerr = str(e)
                QtGui.QMessageBox.warning(self, 'Sorry!', strerr +"\n\n\nSystem could not login. Try again later!")
            except TimeoutError as e:
                strerr = str(e)
                QtGui.QMessageBox.warning(self, 'Sorry!', strerr +"\n\n\nSystem could not login. Try again later!")
            #except:
            #    QtGui.QMessageBox.warning(self, 'Sorry!', "Something went wrong. Try again later!")

            finally:
                self.cursor.close()
                self.db.close()


#send_mail function sends the recovery text from iamnotsparsha@gmail.com to the
    user Email def send_mail(self,userEmail):
        #gmail connection
        try:
            mail=smtplib.SMTP('smtp.gmail.com:587')
            mail.ehlo()
            mail.starttls()
            mail.login('iamnotsparsha@gmail.com', 'fantasy.101')
            content = "Please use this code to log into your account: "
            content = 'Subject: %s\n\n%s' % ("Password Reset", content)
            content=content+ self.code
            mail.sendmail('iamnotsparsha@gmail.com',userEmail,content)
            mail.close()

        except UnboundLocalError as e:
            strerr = str(e)
            QtGui.QMessageBox.warning(self, 'Sorry!', strerr)

        finally:
          pass


    def GetUsername(self):
        #******************first the username is fetched from the Database*************************
        self.uname=str(self.lineEdit.text())
        if not self.uname:
            self.errorlabel.setText("***Enter a username***")
        else:
            try:
```

```
            self.db=mysql.connector.connect(host="localhost", database="python_mysql",user="root",password="2864")
            self.cursor=self.db.cursor()
            self.cursor.execute("select * from registration where username=%s",(self.uname,))
            rows=self.cursor.fetchone()
            if not rows:
               self.errorlabel.setText("***Invalid Username***")
            else:
               '''*************if the username is matched, user details like number of images(inum),
                        tolerance value(tvalue),list of images(imageList) and clickpoints(cpx,cpy)
                        are fetched from the database************************'''
               self.inum=rows[5]
               self.tvalue=rows[6]
               self.imageList=[]                        #list of user images
               self.cpx=[]                          #list of x-coordinates for clickpoints
               self.cpy=[]                          #list of Y-coordinates for clickpoints

               for i in range (self.inum):
                  self.new=json.loads(rows[i+7])
                  self.imageList.append(self.new[0])
                  self.cpx.append(self.new[1])
                  self.cpy.append(self.new[2])
               self.clickpoint.setDisabled(False)
               '''************after username is matched, clickpoint is enabled,
                        correct image is shown and remaining entities are
               closed***********''' self.gencorrectImage()
               self.lineEdit.close()
               self.okay.close()
               self.forgotButton.close()
               self.errorlabel.close()

         except Error as e:
            strerr = str(e)
            QtGui.QMessageBox.warning(self, 'Try again later!', strerr)
         finally:
            self.cursor.close()
            self.db.close()

#if the user clicks on correct series of clickpoints, correct series of images are
   generated def gencorrectImage(self):
      if self.imagenum <self.inum:
         self.loginimage.setPixmap(QtGui.QPixmap(str(self.imageList[self.imagenum])+".png"))

self.clickpoint.setGeometry(QtCore.QRect(self.cpx[self.imagenum],self.cpy[self.imagenum],self.tvalue,self.tvalue))
         self.imagenum = self.imagenum + 1 #increasing the counter else:

         self.imgRetrieve.remove_images()
         self.clickpoint.close()
         self.close()
         self.openHome(self.cpx,self.cpy)
      #self.temp=self.temp+1                    #don't remember what purpose this line of statement served


#f the user clicks on incorrect clickpoints, random series of images are
   displayed def genrandomImage(self):
      self.clickpoint.close()
      self.rand_num=randint(1,12)
      if self.imagenum <self.inum:
         self.loginimage.setPixmap(QtGui.QPixmap(str(self.rand_num)+".png"))
         self.imagenum = self.imagenum + 1 #increasing the counter else:

         self.close()
         QMessageBox.warning(self, 'Sorry!','Incorrect Password. Please try again!')
```

```
#if the user gets logged in then a new window, 'home'
   is shown def openHome(self,cpx,cpy):
      self.homeWindow = Ui_Home(cpx,cpy)
      self.homeWindow.show()



#class Definition for making the label clickable
class ExtendedQLabel(QLabel):

   def __init(self, parent):
      QLabel.__init__(self, parent)

  def mouseReleaseEvent(self,
                                    ev):
self.emit(SIGNAL('clicked()'))

from main_window_ui import *
from registration_ui import *
import sys
import os

if __name__ == '__main__':
   app = QtGui.QApplication(sys.argv)
   mainWindow = Ui_MainWindow()
   mainWindow.show()
   sys.exit(app.exec())
```