Analysis:

I ran both file types the one using salts and the other without using salts. In all cases I used Files that had 1000 users and passwords from 3 to 6 in length. I did not use a binary search tree. I wrote the code in python and ran in in the command line on my machine. I timed how long it took.

5 Runs without using salt took an average of 41.07 seconds to crack all 1000 passwords.
5 Runs using salt took an average of 165.70 seconds

5 runs, no salt 10 passwords = 46.37
5 runs with salt 10 passwords = 58.23 seconds

Doing same thing 100 passwords all length of 4.
With salt = 20.29 seconds
Without salt = 0.77 seconds

It is seeming to me like a salt is nothing more that a password essentially with on extra integer. So let us go ahead and try file type 2 with passwords up to five long to see if it is around 20 seconds.

It took 7.588 seconds so my assumption was definitely wrong.

With that being said. From our measurements it is clear that using a salt is much more secure than not using one. It takes far longer to brute force. In a system of 1000 passwords up to length six our machine can crack the passwords in 41 seconds. The way I ran the code means that per each 1 million possibilities it takes 41 seconds to crack.

For a password to be secure, I am going to say it needs to take more that one week for my system to crack it.

One week is about 604800 seconds. 604800/40 = 15120 this is how many times we can check a new set of 1million passwords in a week. So we get 15120000000 different passwords checked in one week. 10^x > 15120000000 which gives us 11. Our password would have to be 11 in length to be secure. This would actually make is far more secure that one week. Just going from the 10th to the 11th digit would take it from less than one week to crack up to 6.61 weeks to crack.

Code for task1. import hashlib
import sys
import random

```
def verifyUserName(userName):
    file = open("file1","r")
```

```
    if(userName.isalpha()):
        if(len(userName) > 10 or len(userName) < 1):
            print("Username is not an appropriate length. Must be 10 characters long.")
            return False
    else:
        print("Username must be alphabetical characters only.")
        return False

    for line in file:
        uN = line.split(',')
        if(userName == uN[0]):
            print("That username is already taken.")
            return False

    return True

def verifyPassword(password):
    if(password.isdecimal()):
        return True
    else:
        print("Password can be any length but must be all integers.")
        return False

def generateSalt():
    return random.randint(0,9)

def checkFile1(username, password):
    file = open("file1","r")

    for line in file:
        vS = line.split(',')
        if(username == vS[0]):
            if(password == vS[1]):
                print("File 1 login success.")
                return

    print("Improper username or password for file 1")
    return


def checkFile2(username, password):
    file = open("file2","r")
```

```
    tempVal = hashlib.sha256(password.encode())
    hashPass = str(tempVal.hexdigest())

    for line in file:
        vS = line.split(',')
        if(username == vS[0]):

            if(hashPass == vS[1]):
                print("File 2 login success.")
                return

    print("Improper username or password for file 2.")
    return


def checkFile3(username, password):
    file = open("file3","r")

    tempVal = hashlib.sha256(password.encode())
    hashPass = str(tempVal.hexdigest())

    for line in file:
        vS = line.split(',')
        if(username == vS[0]):
            salt = vS[1]
            password += salt

            tempVal = hashlib.sha256(password.encode())
            hashPass = str(tempVal.hexdigest())

            if(hashPass == vS[2]):
                print("File 3 login success.")
                return

    print("Improper username or password for File 3")
    return

# initializing string
again = True

while(again):
    answer = input("Enter c to create and account, l to login, or any other value to quit. ")
    if(answer == 'c'):
        validUser = False
```

```
        validPassword = False

        while(validUser == False):
            userName = input("Please enter a username: ")
            validUser = verifyUserName(userName)

        while(validPassword == False):
            password = input("Please enter a password made of only integers: ")
            validPassword = verifyPassword(password)

        salt = generateSalt()

        outPut1 = open("file1","a+")
        outPut2 = open("file2","a+")
        outPut3 = open("file3","a+")

        outPut1.write(userName+","+str(password)+",\n")

        tempVal = hashlib.sha256(password.encode())
        outPut2.write(userName+","+str(tempVal.hexdigest())+",\n")

        saltedPass = password+str(salt)
        tempVal = hashlib.sha256(saltedPass.encode())
        outPut3.write(userName+","+str(salt)+","+tempVal.hexdigest()+",\n")

        outPut1.close()
        outPut2.close()
        outPut3.close()

    elif(answer == 'l'):
        username = input("Please input your username: ")
        password = input("Please input your password: ")

        checkFile1(username, password)
        checkFile2(username, password)
        checkFile3(username, password)

    else:
        print("Goodbye.")
        exit()
```

Task2.

```
import hashlib
```

```
import sys
import random

def generateSalt():
    return random.randint(0,9)

numOfUsers = int(input("Enter number of users: "))
lowerBound = int(input("Enter lower bound for password length: "))
upperBound = int(input("Enter upper bound for password length: "))


outPut1 = open("file1","w")
outPut2 = open("file2","w")
outPut3 = open("file3","w")

for x in range(0,numOfUsers):

    username = "user"+str(x)
    passLength = random.randint(lowerBound,upperBound)
    password = ""

    for y in range(0,passLength):
        password += str(random.randint(0,9))

    salt = generateSalt()

    outPut1.write(username+","+str(password)+",\n")

    tempVal = hashlib.sha256(password.encode())
    outPut2.write(username+","+str(tempVal.hexdigest())+",\n")

    saltedPass = password+str(salt)
    tempVal = hashlib.sha256(saltedPass.encode())
    outPut3.write(username+","+str(salt)+","+tempVal.hexdigest()+",\n")

outPut1.close()
outPut2.close()
outPut3.close()

Task 3.
import hashlib
import sys
import random
import time
```

```
#Creating this boolean to tell the code whether we are using file type 2 or file type 3
type2 = True
mylist = []

def getPass(list):
    temp = ""
    for x in range(0,len(list)):
        temp+=str(list[x])

    return temp

def checkFile2(password):
    file = open("file2","r")

    tempVal = hashlib.sha256(password.encode())
    hashPass = str(tempVal.hexdigest())

    for line in file:
        vS = line.split(',')
        if(hashPass == vS[1]):
            print(str(password +" is a password found for the username: "+vS[0]))
            return


def checkFile3(password):
    file = open("file3","r")

    tempVal = hashlib.sha256(password.encode())
    hashPass = str(tempVal.hexdigest())

    for line in file:
        vS = line.split(',')


        tempVal = hashlib.sha256(password.encode())
        hashPass = str(tempVal.hexdigest())

        if(hashPass == vS[2]):
            salt = password[len(password)-1]
            password = password[:-1]
            print(str(password +" is a password found for the username: "+vS[0]))
            return
```

```
mylist.append(0)
count = 0
password = ""
begin = time.time()

while(len(mylist)<6):
    password = getPass(mylist)

    #commented out to choose what file type to check.
    checkFile2(password)
    #checkFile3(password)
    mylist[0] += 1

    for y in range(0, len(mylist)):
        if((y + 1) == len(mylist)):
            if(mylist[y] == 10):
                mylist[y] = 0
                mylist.append(0)
        else:
            if(mylist[y] == 10):
                mylist[y] = 0
                mylist[y+1] += 1

end = time.time()
print(end - begin)
```