1.

```
name = str(raw_input("Enter your Username: "))
strData = 'Hello <<UserName>>, How are you?'
if len(name) >= 3 : # and name.isalnum():
    StrData = strData.replace('<<UserName>>', name[:-1])
    print strData
else:
    print("Use a valid username")
```

2.

```
import time

print''' input as needed:\n Type 1 to start the stopwatch\n
                              Type 2 to End the stopwatch '''


def time_convert(sec):
mins = sec // 60
sec = sec% 60
hours = mins// 60
mins = mins % 60
print("Time Lapsed = {0}:{1}:{2}".format(int(hours), int(mins),
                                                    sec))


flag = True
while flag
        choice = input("Input : ")
        if choice == 1 or choice == '1':
            start = time.time()
            print ("Type 2 to stop stopwatch \n Type 3 to Exit")
```

```
choice = input("Input in :")
end = time.time()
if choice == 2 or choice == '2':
    time_elapsed = end - start
    time_convert(time_elapsed)
else:
    print 'closing..."
    exit()
else :
    print "closing..'
    exit()
```

3.

```
import random
import sys

board = [i for i in range(0,9)]
player, computer = '', ''
moves = ((1,7,3,9),(5,),(2,4,6,8))

winners = ((0,1,2),(0,3,6),(0,4,8),(1,4,7),(2,4,6),
           (2,5,8),(3,4,5),(6,7,8))

tab = range(1,10)

def boardShow():
    x = 1
    counter = 0
    for i in range(0,9):
        if board[i] == 'x' or board[i] == 'O':
            print(board[i], end = ' ')

    else:
        print(" ", end = ' ')
    counter += 1
    if counter % 3 == 0 :
        print("|", end = ' ')

def select_char():
    if random.randint(0,1) == 0:
        return ('x','O')
    return ('O','x')
```

```
def can_move(brd, player, move):
    if move in tab and brd[move-1] == move-1:
        return True
    return False

def can_win(brd, player, move):
    places = []
    x = 0
    for i in brd;
        if i == player: places.append(x);
        x += 1
    win = True
    for tup in winners:
        win = True
        for ix in tup:
            if brd[ix] != player:
                win = False
                break
        if win == True
            break
    return

def make_move(brd, player, move, undo = False):
    if can_move(brd, player, move):
        brd[move-1] = player
        win = can_win(brd, player, move)
        if undo
            brd[move-1] = move-1
        return (True, win)
    return (False, False)
```

```
def random_move():
    move = -1
    empty_place = []
    for i in range(1, 10):
        if board[i-1] == i-1:
            empty_place.append(i)
    if len(empty_place) > 0:
        idx = random.randint(0, len(empty_place)-1)
        move = empty_place[idx]
        return make_move(board, computer, move)
    return (False, False)

def machineMove():
    move = -1
    for make_move(board, computer, i, True)[1]:
        move = i
        break
    if move == -1:
        for i in range(1, 10):
            if make_move(board, player, i, True)[1]:
            move = i
            break
    if move == -1:
        for tup in moves:
            for mv in tup:
                if move == -1 and can_move(board, computer, mv):
                move = mv
                break
    return make_move(board, computer, move)
```

```
def is_full():
    for i in range(0,9):
        if board[i]==i:
            return True
        return False
player, computer = select_char()
print('player is [%s] and computer is [%s]'%(player, computer))

turn = random.randint(0,1)
if turn==0:
    print("player will play first.")
else:
    print("computer will play first.")
    boardShow()
    machineMove()
    print()
result = "It's a tie!!"
while is_full():
    boardShow()
    print('Make your move [1-9]: ', end='')
    move = int(input())
    moved, won = make_move(board, player, move)
    if not moved:
        print('>> Invalid number! Try again!')
        continue
    if won:
        result='you won!!'
        break
    elif machineMove()[1]:
        result='you lose!!'
        break;
boardShow()
print(result)
```

4.
```
NO_OF_CHARS = 256
str = ''

def isPalindromePossible(st):
    count = [0] * (NO_OF_CHARS)
    for i in range(0, len(st)):
        count[ord(st[i])] = count[ord(st[i])] + 1

    odd = 0
    for i in range(0, NO_OF_CHARS):
        if (count[i] & 1):
            odd = odd + 1
        if (odd > 1):
            return False
    return True

def isPalindrome(st):
    if (st == st[::-1]):
        return True
    else:
        return False
data = input("Enter a string:")
if (isPalindromePossible(data)):
    print('palindrome possible.')
    if isPalindrome(data):
        print(data)
else:
    palList = []
    palList = set(data)
```

```
    revStr = str.join(palList)
    freshStr = revStr[::-1] + revStr
    print(freshStr)
else
    print('palindrome not possible.')
```

5. a.
```
def reverse(self):
    prev = None
    current = self.head
    while(current is not None):
        next = current.next
        current.next = prev
        prev = current
        current = next
    self.head = prev
```

<Reverse using recursion>

b.
```
def rereverseUtil(self, curr, prev):
    if curr.next is None :
        self.head = curr
        curr.next = prev
        return
    next = curr.next
    curr.next = prev
    self.reverseUtil(next, curr)

def reverseRecusion(self):
    if self.head is None :
        return
    self.reverseUtill(self.head, None)
```

<Reverse using stack>

c.
```
def reverseStack(self, head):
    stk = [] # a stack basically print by -1
    ptr = head
    while (ptr != None):
        stk.append(ptr)
        ptr = ptr.next
    return stk
```