1.

Chiho Li

64011378

```
def find_word_positions(word,list_of_word):
    holder = []
    converted = word.lower()
    num = len(list_of_word)

    for x in range(num):
        if converted == list_of_word[x].lower():
            holder.append(x)
    if holder ==[]:
        return 0
    else:
        return holder
```

2.

```
popularity_scores={"c++": 99.7,
                   "c  ": 96.5,
                   "Java": 97.5,
                   "Python": 100;
                   "c#": 89.4}
def ranking_score(usin)
    holder = []

    for x in usin:
        if usin[x] not in holder:
            holder.append(usin[x])
    holder.sort()
    print(holder)
print(ranking_score(popularity_scores)
```

4. Saving Account Chiho Li 64011378

```
class SavingAccount:
    def __init__(self, bank_name, acc_name, acc_id, balance):
        self.bn = bank_name
        self.an = acc_name
        self.ai = acc_id
        self.bal = balance
        self.his = []
    def deposit(self, money, person, date):
        self.bal += money
        print(f"you have successfully deposited {money} baht.")
        print(f"Task completed on {date}, {person}!)
        self.his.append(f"Deposit {money} baht into {self.bn}, ID:{self.ai},
        Date: {date}, Name: {person}")
    def withdraw(self, money, person, date):
        if ((self.bal - money < 0):
            Print("you do not have enough money!, get richer")

        else:
            self.bal -= money
            print(f"you have successfully withdrawed {money} baht.")
            print(f"Task completed on {date}, {person}!)
            self.his.append(f"Withdraw {money} baht from {self.bn}, ID:{self.ai},
            Date: {date}, Name: {person}")
def get_balance(self):
    print(f"{self.an}'s current balance: {self.bal}")
def print_statement(self):
    for audit_log in self.his:
        print(audit_log)
```

Chiho Li

64011378

```
chiho = SavingAccount("kbank", "Chiho", 69, 5000
chiho. deposit (500, "chiho", "May 18")
Chiho. withdraw (5000, "chiho", "June 14")
chiho. withdraw (5000, "chiho", "June 16")
chiho. get_balance ()
Chiho. print_statement()
```

/Over Drawn Account Part II

```
class OverDrawnAccount (SavingAccount)
    def __init__ (self, bank_name, acc_name, acc_id, balance, limit):
        super().__init (bank_name, acc_name, acc_id, balance)
        self.his = []
        self.limit = limit
    def deposit (self, money, person, date):
        super().deposit(money, person, date)
    def withdraw (self, money, person, date):
        if ((self.bal - money) < self.limit):
            Print(f"sorry, you are exceeding the limit!")
        else:
            self.bal -= money
            print (f"you have successfully withdrawed {money} baht.")
            print (f"Task complete on {date}, {person}!"}
            self.his.append (f"withdraw {money} baht from {self.bn}, ID: {self.ai},
            Date: {date}, Name: {person}")
    def get_balance (self):
        super(). get_balance ()
    def print_statement(self):
        super(). print_statement
```

5.

```
import ABC

class Sale_item (abc.ABC)
    def __init__(self, amount, price):
        self. amount = amount
        self. price = price

    @ abc.abstractmethod
    def total_cast(self):
        pass

class Food (sale_item):
    def __init__ (self, amount, price ):
        super(). __init__ (amount, price)

    def cost (self):
        print (self. amount * self. price)


class Measured_food (Food):
    def __init__ (self, weight, price ):
        super(). __init__ (weight, price )
        self. weight = weight
    def cost (self)
        print ( self. weight * self.price)


class Itemized_food (Food):
    def __init__ (self, amount, price)
        super (). __ Init__ (amount, price )
    def cost (self):
        print ( self.amount * self. price )
```

```
class Appliance ( sale_ item ):
    def __ init_ (self, amount, price ):
        super(). __ init__ (amount, price )

    def cost (self)
        print (1.07 * (self.amount * self.price))

class Book (sale_item ):
    def __init__ (self, amount, price ):
        super(). __ init__ (amount, price)

    def cost (self):
        discounted = self. price * 0.07
        print ((self. price * self. amount ) -
               discounted ))
```