Câu 1:

1. Compute the number of possible testcases:

• For Customer name: 6 cases

o <Empty>

o User enter a number

o User enter a special character (\*&^%$...)

o Number of characters less than 2

o Number of characters greater than or equal 2 and less than or equal 24

o Number of characters greater than 24

• For account number: 6 cases

o <Empty>

o The string consist of non-numeric character(s)

o The string has less than 6 digits

o The string has greater than 6 digits

o The string has exactly 6 digits and 1st is zero

o The string has exactly 6 digits and 1st isn’t zero

• For Loan amount requested: 7 cases

o <Empty>

o User enter non-numeric character(s)

o The value is less than 500

o The value is 500

o The value is greater than 500 and less than 9000

o The value is 9000

o The value is greater than 9000

• For term of loan: 7 cases

o <Empty>

o User enter non-numeric character(s)

o The value is less than 1

o The value is 1

o The value is greater than 1 and less than 10

o The value is 10

o The value is greater than 10

• For monthly repayment: 5 cases

o <Empty>

o User enter non-numeric character(s)

o The value is less than 100

o The value is 100

o The value is greater than 100

=> Total of test cases is: 6 \* 6 \* 7 \* 7 \* 5 = 8820 (testcases)

2. Assume that we need 1 seconds for each testcase, so the total time needed to test all these test cases are 8820 seconds = 147 minutes = 2.45 hours

3

Testing principles:

* Testing shows the presence of defects.
* Exhaustive testing is impossible.
* Early testing.
* Defect clustering.
* Pesticide paradox.
* Testing is context dependent.
* Absence-of-errors fallacy.

Ex:

- Kiểm tra sớm giúp tiết kiệm thời gian và tiền bạc: trong một chương trình, việc kiểm tra sớm sẽ giúp chúng ta biết phát hiện được khuyết tật và việc sửa đổi cũng như bổ xung sẽ trở nên dễ dàng và ít chi phí hơn khi ta đã hoàn chỉnh chương trình.

- Pesticide paradox: nghịch lý thuốc trừ sâu: hiểu đơn gian thì cũng như cái tên của nó, nếu dùng thuốc trừ sâu nhiều lần lên cây thì công trùng sẽ sinh ra miễn dịch, điều này tương tự như test một chương trình nhiều lần bằng một phương pháp giống nhau thì sau nhiều lần kiểm thử, chúng ta không tìm ra vấn đề mới.

Câu 2:

1

hàm max2.py

def max2(*x*,*y*):

           if  *x* >= *y*:

               return *x*

           else:

               return *y*

max2(14,3)

hàm maxSequence.py

def maxSequence(*s*):

    max = *s*[0]

    for elem in *s*:

      if elem > max:

         max = elem

    return max

maxSequence([1,3,4,7])

2

Hàm funct(x,y)

def funct(*x*,*y*):

    if (0<*x*<=1) and *y*<0 :

        return *x*+*y*

    if (*x*>1) and (0<*y*<=5) :

        return *x*\**y*

    else:

        return *x*-*y*

funct(1/2,-2)

funct(2,4)

funct(1,5)

Câu 3:

Câu 4:

def  puschaseAmount(*x*):

    if (999<=*x*<1999) :

        return "5%"

    if (1999<=*x*<3999) :

        return "10%"

    if (3999<=*x*<5999) :

        return "15%"

    if (5999<=*x*<7999)  :

        return "25%"

    if (7999<=*x*<9999) :

        return "35%"

    if (*x*>=9999) :

        return "50%"

    else :

        return "0%"

# puschaseAmount(100)

# puschaseAmount(1000)

# puschaseAmount(5000)

puschaseAmount(7000)

# puschaseAmount(9000)

# puschaseAmount(10000)

Câu 5:

def discountComput(*x*,*y*):

    if (*x*>500) and *y*=="yellow card" :

        return "Discounted 6%"

    if (*x*>500) and *y*=="white one" :

        return "Discounted 5%"

    if (300<*x*<=500) and *y*=="yellow card" :

        return "Discounted 4%"

    if (300<*x*<=500) and *y*=="white one" :

        return "Discounted 3%"

    if (200<=*x*>=300) and *y*=="yellow card" :

        return "Discounted 3%"

    if (200<=*x*>=300) and *y*=="white one" :

        return "No discount."

    else:

        return "No discount."

# discountComput(550,"yellow card")

# discountComput(550,"white one")

# discountComput(400,"yellow card")

# discountComput(400,"white one")

# discountComput(250,"yellow card")

# discountComput(250,"white one")

# discountComput(150,"yellow card")

# discountComput(150,"white one")