CSEN 102 - Midterm 2014 solutions

November 12, 2015

1 Sequential ALgorithms - Consumer Price Index

The formula for calculating the Inflation Rate using the Consumer Price Index (CPI) is relatively simple. Assume for the sake of simplicity that the index consists of one item and that in 1984 that item cost 100 dollars. In December of 2014 that same item would probably cost 198 dollars. Let us calculate the price difference between 1984 and 2014.

- Step 1: Calculate How Much has the Consumer Price Index Increased? By looking at the above example, common sense would tell us that the index increased (it went from 100 to 198). The question is how much has it increased?To calculate the change we would take the second number (198) and subtract the first number (100). The result would be 98. So we know that from 1984 until 2014 prices increased (Inflated) by 98 points.
- Step 2: Comparing the CPI Change to the Original CPI Since we know the increase in the Consumer Price Index we still need to compare it to something, so we compare it to the price it started at (100). We do that by dividing the increase by the first price or 98/100. the result is (.98).
- Step 3: Convert it to a Percentage This number is still not very useful so we convert it into a percentage. To do that we multiply by 100. So the result is a 98% increase in prices since 1984.

Write an algorithm that given two prices of the same item in two different years computes the CPI. The algorithm should display the following for the example above

The price of the item in year 1984 is 100 The price of the item in year 2014 is 198 The CPI is 98%

```
firstYear  = eval(input())
secondYear = eval(input())
oldPrice  = eval(input())
newPrice = eval(input())

CPI = newPrice - oldPrice
CPI = CPI / oldPrice
CPI = CPI * 100
print("The price of the item in year", firstYear , "is", oldPrice)
print("The price of the item in year", secondYear, "is", newPrice)
print("The CPI is", CPI, "%")
```

2 Conditional Algorithms - Refactoring

Given the following algorithm

```
grade = input()
if grade == 'A':
  print(85)

if grade == 'B':
  print(67)


if grade == 'C':
  print(50)

if grade == 'F':
  print(30)

if not (grade == 'A' or grade == 'B' or grade == 'C' or grade == 'F'):
  print(-1)
```

(a) What are the main drawbacks (disadvantages) of the this algorithm?

Solution

All conditions will be checked even if we do not need to check them. Moreover, the last condition includes many checks that can be removed if nested-if statements are used or else if (elif) statements.

(b) Rewrite this algorithm to avoid the drawbacks you listed in part a).

Solution

```
grade = input()
if grade == 'A':
  print(85)
elif grade == 'B':
  print(67)
elif grade == 'C':
  print(50)
elif grade == 'F':
  print(30)
else:
  print(-1)
```

3 Conditional Algoriths - The sum of rounded numbers

Write an algorithm that given three integers displays the sum of their rounded values. We will round an integer value to the next multiple of 10 if its rightmost digit is 5 or more, so 15 rounds up to 20. Alternately, round down to the previous multiple of 10 if its rightmost digit is less than 5, so 122 rounds down to 120. Here are some examples of evaluating the algorithm on representative input values:

```
16 17 12 -> 50
12 13 14 -> 30
6 4 4 -> 10
```

```
a = eval(input())
b = eval(input())
c = eval(input())

aa = a % 10
bb = b % 10
cc = c % 10

if aa >= 5:
  a = 10 + a - aa
else:
  a = a - aa

if bb >=5:
  b = 10 + b - bb
else:
  b = b - bb

if cc >= 5:
  c = 10 + c - cc
else:
  c = c -cc

sum = a + b + c

print(sum)
```

4 Iterative Algorithms - 362 pattern

(a) Write an algorithm that given a list of integers displays true if the list contains a 3, 6, 2 pattern; that is, a value, followed directly by the value plus 3, followed directly by the value minus 1. Here are some examples of evaluating the algorithm on representative input values:

```
{1, 3, 6, 2} -> true
{1, 2, 7, 1} -> false
{3, 6, 2} -> true
{4, 7, 3} -> true
{5, 8} -> false
{1, 3, 6, 2, 5, 1} -> true
```

Your algorithm should stop whenever the pattern is found.

Solution

```
n = int(input())

i = 0
a = [ ]
# Read values from user
while i < n:
  prompt = "Enter element %d" % i
  # prompt = Enter element 0
  # prompt = Enter element 1
  # ...
  value = eval(input(prompt))
  a.append(value)
  i = i + 1

flag = False
if n >= 3:
  i = 0
  while i < n - 1 and not flag:
    if a[i + 1] - a[i] == 3 and a[i + 2] - a[i] == -1:
      flag = True
    i = i + 1

print(flag)
```

(b) Write an algorithm that given a list of integers displays the number of 3, 6, 2 pattern in the list. Here are some examples of evaluating the algorithm on representative input values:

```
{1, 3, 6, 2, 5, 1} -> 2
{1, 2, 7, 1} -> 0
{3, 6, 2, 5} -> 1
{4, 7, 3} -> 1
```

Solution

```
n = int(input())

i = 0
a = [ ]
# Read values from user
while i < n:
  prompt = "Enter element %d" % i
  # prompt = Enter element 0
  # prompt = Enter element 1
  # ...
  value = eval(input(prompt))
  a.append(value)
  i = i + 1

count = 0
if n >= 3:
  i = 0
  while i < n - 1:
    if a[i + 1] - a[i] == 3 and a[i + 2] - a[i] == -1:
      count = count + 1
    i = i + 1

print(count)
```

5 Iterative Algorithms - Deletion of Bad Pairs

(a) Write an algorithm that takes a list of numbers and removes any adjacent pair of integers in the list if the left element of the pair is larger than the right element of the pair. Please note that every pairs left element is stored in an odd-numbered index in the list, and every pairs right element is stored in an even-numbered index in the list. For example, suppose a list A stores the following element values:

```
[3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1]
```

the algorithm should display the following sequence:

```
3 7 5 5 4 7
```

Solution

```
n = int(input())

i = 0
a = [ ]
# Read values from user
while i < n:
   prompt = "Enter element %d" % i
   # prompt = Enter element 0
   # prompt = Enter element 1
   # ...
   value = eval(input(prompt))
   a.append(value)
   i = i + 1

i = 0
while i < n - 1:
   if a[i]  <= a[i + 1]:
      print(a[i])
      print(a[i + 1])
   i = i + 2
```

(b) Assume that the elements of the list are represented now as an integer number. For example, the list

```
[3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1]
```

is now represented as the integer number

```
37925585634731
```

You are asked to perform the same task described in Part a) however this time you are not allowed to use lists. You have to work on the integer number and your algorithm should display for the number above the following result:

```
375547
```

Note that the order is important. You are not allowed to display 745573 and **you are not allowed to use neither lists nor strings at all.** For simplicity, assume that the number consists of an even number of digits.

Solution

```
n = eval(input())
i = 1
j = 10
result = 0
# Hint:
# 123456 % 10 is 6
# 123456 % 100 is 56
# the integer division of 123456 by 10 is 12345 (one digit removed)

while n > 0:
  n1 = n % 1−
  n = n/10
  n2 = n % 10
  n = n/10
  if n1 >= n2:
    result = result + (n2 * j) + (n1 * i)
     i = i * 100
     j = j * 100

print(result)
```

6 Tracing Iteration - Mysterious task

Given the following algorithm

```
m = int(input())
i = 0
a = [ ]
while i < m:
  value = eval(input())
  a.append(value)
  i = i + 1

n = int(input())
b = [ ]
i = 0
while i < n:
  value = eval(input())
  b.append(value)
  i = i + 1

i = 0
j = 0
flag = True

while i <= n and j <= m and flag:
  if a[j] < b[i]:
    j = j + 1
  elif a[j] == b[i]:
    i = i + 1
    j = j + 1
  elif a[j] > b[i]:
    flag = False

if i <= n or not flag:
  print("Mysterious No")
else:
  print("Mysterious Yes")
```

(a) What is the output of the algorithm above for the following two lists:

```
List A: 1 2 5 8 44
List B: 2 6 8
```

Draw a tracing table.

Solution

i	j	flag	a[j]	b[i]
0	0	True	1	2
0	1	True	2	2
1	2	True	5	6
1	3	True	8	6
1	3	False		

```
Mysterious No
```

(b) What does the algorithm do for any two sorted (in increasing order) lists A and B, i.e. what is the meaning of "Mysterious Yes" and "Mysterious No"?

Solution

The algorithm checks whether the list a is a subset of list A. Thus, it will print "Mysterious Yes" if list b is a subset of list A and prints "Mysterious No" otherwise.

(c) Try to execute the algorithm on the following lists

```
List A: 1 2 5 8 44
List B: 1 2 5 8 44 100 202
```

According to the functionality of your algorithm, this case can be handled in a more efficient way. Add the corresponding statements in the code below for the case where the length of list B is greater than the length of list A.

Solution

```
m = int(input())
i = 0
a = [ ]
while i < m:
  value = eval(input())
  a.append(value)
  i = i + 1

n = int(input())
b = [ ]
i = 0
while i < n:
  value = eval(input())
  b.append(value)
  i = i + 1

i = 0
j = 0
flag = True

if m > n:
  while i <= n and j <= m and flag:
    if a[j] < b[i]:
      j = j + 1
    elif a[j] == b[i]:
      i = i + 1
      j = j + 1
    elif a[j] > b[i]:
      flag = False

if i <= n or not flag:
  print("Mysterious No")
else:
  print("Mysterious Yes")
```