CGS600A: Computational Tools in Cognitive Science

Quiz #1

Max marks: 40
Time:30 minutes 3 Oct. 2018

1. *You can only refer to your own handwritten class notes and nothing else.*
2. *Where needed assume that code fragments are typed into the Python interpreter.*
3. *Any code is shown using typewriter font like* `this`.
4. *If you think executing the code will lead to an error or an exception then indicate the error/ exception.*

1. The class `Sort` (see code below) sorts a list of entities that can be compared using a compare function. Sorting is being done using a merge-sort method. The `merge` operation merges two sorted lists to produce a single sorted list. And the `sortfn` sorts the list passed as argument using the comparison function `_cmpfn` available in the class.

   (a) Some code indicated by `??1??` to `??8??` is missing. Fill in the missing code so that the class functions correctly.

   Solution:

   ```
   ??1?? is lambda x, y: x<=y
   ??2?? is self._cmpfn(l1[0], l2[0])
   ??3?? is l.append(l1.pop(0))
   ??4?? is l=l+l2
   ??5?? is l=l+l1
   ??6?? is l[:n//2]
   ??7?? is l[n//2:]
   ??8?? is sortfn(self.getlst())
   ```

   (b) Assuming the class `Sort` has been read into the interpreter write the statement(s) you will type in the interpreter that will allow you to sort in descending order the contents of list `L` that has comparable elements. How will you confirm that `L` has been sorted?

   Solution:

   ```
   so=Sort(L, lambda x, y: x>y)
   so.sort()
   so.getlst() # Used to check if the list has been sorted.
   ```

```
class Sort:
    def __init__(self, l=[], f=??1??):
        self._lst=l
        # Default comparison function sorts in ascending order
        self._cmpfn=f #comparison function
```

```
def getlst(self):
    return self._lst

def setlst(self, l):
    self._lst=l

def setcmpfn(self, f):
    self._cmpfn=f

def sort(self):
    def merge(l1, l2):
    # merges two sorted lists to produce single sorted list.
        l=[]
        while l1!=[] and l2!=[]:
            if ??2??:
                ??3??
            else:
                l.append(l2.pop(0))
        if l1==[]:
            ??4??
        else:
            ??5??
        return l

    def sortfn(l):
        n=len(l)
        if n<2:
            return l
        else: #merge two approx. equal sized sorted lists
            return merge(sortfn(??6??), sortfn(??7??))

    self.setlst(??8??)
```

[4×8,(6,2)=40]