Chennai Mathematical Institute

Advanced Programming

Final Examination, Summer Semester, 2020–2021

Date : 30 July, 2021 (2:00pm-5:00pm) Marks : 70
Duration : Three hours + 30 minutes (for scanning) Weightage : 35%

1. Assume we have a Python class `List` in which each node has the basic structure indicated by the definition to the right. The value is never `None` except for an empty list. The last element in the list has `next` equal to `None`.

   ```
   class List:
     def __init__(self):
       self.value = None
       self.next = None
       return
   ```

   Add a Python function to the class that takes as argument another `List` and merges the second list with the original one assuming both the lists are sorted in non-decreasing order. In other words the call `lst1.merge(lst2)` should update `lst1` with the merge of the two sorted lists `lst1,lst2`. *(10 marks)*

2. Suppose you are given a directed graph as a (Python) list of tuples each of which consists of:

   (a) A number from 1 to n (representing a vertex u)

   (b) A value which is either an integer or one of the operands '+','*'.

   (c) A list of numbers from 1 to n (representing the list of out-neighbours v of the the above vertex u i.e. (u, v) is a directed edge of the directed graph)

   Your first task is to write a Python program to verify that there is a single source (i.e. vertex of indegree zero) in the graph and that the values corresponding to all sinks (i.e. vertices of out-degree zero) are integers while values of non-sinks are operands from '+','*'.

   Your second task is to write a Python program to verify that the graph is a DAG (directed acyclic graph).

   Your third task is to write a Python program that computes the integer at the source node assuming that the graph satisfies the above two properties. Notice that the integer at a vertex is recursively defined as its integer value if its a sink and as the integer obtained by applying the operand '+' or '*' according to its value, to the integers computed by its out-neighbours. Here '+','*' are interpreted as the usual arithmetic operations.

   Your Python code should run in time proportional to the total length of all the lists given to you. Assume that common operations like list indexing, addition of two integers and multiplication of two integers takes $O(1)$ time.

   While the emphasis of this question is on writing an algorithm, you are encouraged to stick as closely as possible to writing valid Python code. The Python procedures should be self contaiend and not depend on any libraries. *(14 marks)*

3. Suppose you have a function that given a sequence of integers and another integer s answers if there is a subsequence that sums to exactly s. Your goal is to identify such a subsequence (if it exists) from a sequence of n integers and given the desired sum s by using $O(n)$ calls to the function. *(11 marks)*

4. Suppose you have a square board consisting of n rows and n columns so that there are n^2 cells in the board. You are allowed to place a coin anywhere in the southmost row of the board. At every step you have to move the coin to the row immediate north of the current row in one of the at most three cells:

- The cell directly north of the current cell
- The cell that is one cell to the east of the cell directly north (if defined)
- The cell that is one cell to the west of the cell directly north (if defined).

You earn (possibly a negative number) $p(c_1, c_2)$ when you move from cell c_1 to cell c_2 in one step according to the above rules. The aim is to pick a starting cell in the southmost row and a sequence of moves leading to a cell in the northmost row that gurantees the maximum overall profit which is the sum of the $p(c_1, c_2)$ along the moves. The table p is provided with values for pairs representing valid moves according to the above rules. Write an algorithm (not Python code) to achieve this and compute its time complexity. *(12 marks)*

5. Fix integers k, n such that $k \leq n$. Let $G = (V, E)$ be a graph where $V = \{1, \dots, n\}$ and $E = \{\{i, j\} : i + j \equiv 0 \bmod k$. Notice that E might include self loops. A 1-factor of G is a subset of edges of E such that each vertex is incident on exactly one edge of E (possibly a self-loop). Show that you can compute a 1-factor of G in $O(n)$ time. *(15 marks)*

6. Assume that we have defined a class `Node` to build a linked list, and a function `insert(l,n)` that inserts a new value `n` at the head of the list `l`, as given below. The function `new(Node())` allocates a fresh object of type `Node` and returns a reference (pointer) to the location of this object.

```
class Node {
  int value;
  Node next;
  ...
}
```

```
boolean insert(Node head, int newvalue){
  Node newnode;
  newnode = new(Node());
  newnode.next = head.next;
  newnode.value = head.value;
  head.next = newnode;
  head.value = newvalue;
  return(True);
}
```

Let S and H be the memory stack and heap, respectively, before a call to `insert(..)`. Using diagrams, explain how the stack and heap are transformed with respect to the original S and H when `insert(..)` is running, and after `insert(..)` returns.

(8 marks)