Question	1 (5 marks)	2 (5 marks)	3 (6 marks)	4 (4 marks)	Total (20 marks)
Marks	5	4	6	4	19

Please write your name on every page and enter your serial number in the box above. **If you miss out any of these: -1 and no rechecking.**

No pseudocode means: For every loop you use, you must explain what it will do to the input. You cannot write things like "x = x + y", you must explain the significance of every step in words. You cannot write "for i = 1 to ..." or "while <condition>...", you must *explain* what the loop achieves. In summary: if we need to interpret how your description will treat a particular input then it is pseudocode.

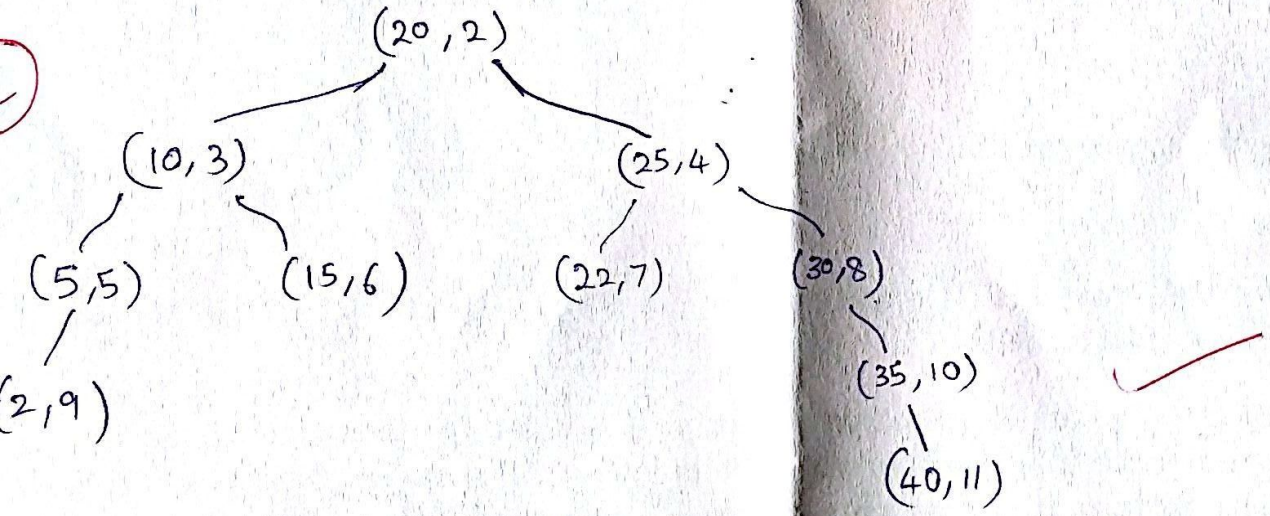
Q1. (Tree + heap = Treap. Total marks = 5). Marks: 5

Let us suppose we are given a set S of the following 10 items, each is a tuple with two values, the first a key, the second a priority:

$$S = \{(22,7),(20,2),(2,9),(10,3),(35,10),(40,11),(15,6),(30,8),(25,4),(5,5)\}.$$

A *Treap* is a special kind of binary search tree which also has the property of a heap. Each item in a treap is a tuple of two values i.e. $x_i = (k_i, p_i)$ where k_i is a key and p_i is a priority. A treap is a binary search tree on the keys of the items and maintains the *min-heap order* property on the priorities i.e. the priority value of a node is less than the priority value of any children it may have. Note that the Treap *does not* have to have the structural property of heap i.e., it may not be a complete binary tree but must have the order property of heap.

Q1.1 (2 marks). Construct a Treap on the set S given above. You do not need to show the steps, only the final treap.



Name:_____ Entry No:_____

Q1.2 (3 marks). Given a set of key-priority pairs, like S above, give an algorithm to construct a treap for S. Note, you are to assume the entire set is given in advance. **No pseudocode.**

Since, the set is given we can traverse it to find element e_1 with min priority. This is the root of our tree. Let $e_1 = (k_1, p_1)$ where $p_1 < p_i$'s of other elements of set.

The remaining elements of the set are split into two group : Group 1 : Elements with keys $\leq k_1$
Group 2 : Elements with keys $> k_1$

We recurse (Treap construct) on (Group 1) - this is left subtree
and on (Group 2) - right subtree

Treapconstr (Group 1) is left subtree of root e_1

Treap construct (Group 2) is right subtree of root e_1

3

Marks: 4

Q2. (Hastables. Total marks = 5)
In this question we look at an open addressing scheme called *cuckoo hashing*. The set of keys we are trying to store is a subset of the natural numbers and we have a hash table T of size n. We are given two hash functions h_1 and h_2 which map all natural numbers to the set $\{0, 1, \ldots, n-1\}$. In order to place a newly inserted key x into the table we do the following:

1. Compute $l_1 = h_1(x)$.
2. If $T[l_1]$ is free then store x in $T[l_1]$ and exit, else
3. Compute $l_2 = h_2(x)$.
4. If $T[l_2]$ free then store x in $T[l_2]$ and exit else if y is currently stored in $T[l_2]$. Remove y from $T[l_2]$ and store x there.
5. Now we need to find an alternate position for y. If l_2 was $h_1(y)$ then compute $l_3 = h_2(y)$ and go to step 4 else if l_2 was $h_2(y)$ then compute $l_3 = h_1(y)$ and go to step 4. The fifth time we have to go to step 4 we assume a cycle has occured and an exception is thrown declaring the insertion unsuccessful.

Q2.1. **(1 mark)** Given a key x, explain in words how to check if x is in the hash table.

~~Check for x at $h_1(x)$ then $h_2(x)$ then $h_1(y)$~~

Check for x at $h_1(x)$, $h_2(x)$, $h_1(z)$, $h_2(z)$, ~~$h_1(y)$, $h_2(y)$~~

simply check $h_1(x)$ and $h_2(x)$

Here if y is at $h_1(x)$ and z is at $h_2(x)$

0

58		89		69		49		18		
0	1	2	3	4	5	6	7	8	9	10

Figure 1: A cuckoo hash table with $n = 11$ and the keys 69, 58, 49, 18 and 89 inserted .

Q2.2. **(4 marks)** Suppose we are given $n = 11$ and $h_1 = (x + 1) \bmod n$ and $h_2 = (2x + 5) \bmod n$ and the following sequence of insertions starting from an empty table: 69, 58, 49, 18, 89, 55, 56, 28 is performed. The table after the insertions of 69, 58, 49, 18, 89 is shown in Figure 1. Show the state of the hash table after *each* of the following insertions 55, 56 and 28 using cuckoo hashing. Note you *only have to show the state of the array at the very end of each operation, not the intermediate movements.* In case you detect a cycle (i.e. you throw an exception and abort the insertion) please show the state of the hashtable at the 5th time the "go to" statement is executed and indicate the value y which was last displaced.

$h_1(55) = 1$, ~~$h_2(55) = 6$~~

4

58	55	89	69	49	18					
0	1	2								

$h_1(56) = 2$ occupied ; $h_2(56) = 7$

58	55	89		69		49	56	18		
0	1	2	3	4	5	6	7	8	9	10

$h_1(28) = 6$; $h_2(28) = 6$

Remove 49

69	55	89		58		49	56	18		
0	1	2	3	4	5	6	7	8	9	10

Exception : can't insert 28

last displaced element was 28 by 49

Q3. (Binary Search Trees. Total marks = 6) We have a Binary Search Tree implementation in which each node contains an extra value along with the key that we call **special**. This is some pseudocode for insertion into this BST:

6

```
function insert(T,x)
   temp = CreateNode(x)
   if (T is empty)
   {
     Set temp.special = 1
     Set root of T to be temp
   }
   else
   {
     curr = getRoot(T)
     Set curr.special = (curr.special + 1) mod 2
     if (x <= curr.key)
     {
       if curr.leftSubtree.isEmpty()
       {
         Set left child of curr to be temp
       }
       else
       {
         insert(temp.leftSubtree,x)
       }
     }
     else
     {
       if curr.rightSubtree.isEmpty()
       {
         Set right child of curr to be temp
       }
       else
       {
         insert(temp.rightSubtree,x)
       }
     }
   }
```

Q3.1. (1 mark) What does the **special** value stored at a node represent?

Special value of node is 1 if node has even no. of children and 0 if node has odd no. of children.

1