Midterm 2 Grading Guide

The purpose of the scoring guide is to provide a uniform evaluation of the tests by the graders. Therefore, the guide gives you the main ideas (and at least one possible solution) for each problem and the associated scores. This guide is not intended to be a detailed description of a complete solution to each problem; the steps described are sketches of a solution that achieves maximum score.
The partial scores given in the guide are only awarded to the solver if the related idea is included in the test as a step in a clearly described and reasoned solution. Thus, for example, the mere description of statements or definitions in the material without its application (even if any of the facts described do indeed play a role in the solution) is not worth the point.
A partial score is given for any idea or part solution from which the correct solution of the problem could be obtained with the appropriate addition of the reasoning described in the test. If a solver starts several solutions that are significantly different, then you can give score for at most one. If each described solution or part of solution is correct or can be extended to be correct, then the solution with the most points will be evaluated. However, if there are correct and incorrect solutions between several solution attempts and it is not clear from the test which one the solver considered to be correct, then the solution with fewer points will be evaluated (even if this score is 0). The scores in this guide may be further subdivided as necessary. Of course, a good solution other than that described in this guide is also worth the maximum score.
In the event of an arithmetic error, one point is deducted from each exercise. An exception to this is when computing the numbers significantly simplifies or modifies the structure of the task. In these cases, no points are awarded to the steps which follow the computation and are missing from the solution of the student.

---

1. The array of a heap contains the following elements (in this order): 3, 5, y, x, 12, 10, 100, 7, 16, 21.
   (a) Draw the heap as a binary tree.
   (b) After one DELETEMIN operation, y becomes the root. Draw the heap after this operation as a binary tree.
   (c) Determine all possible values of x and y, if we know that the heap contained all distinct numbers (and that y is the root after a DELETEMIN operation on the original heap).

   Grading guide:

   (a) Correct drawing of tree: **2 points**
   (b) Correct drawing of the tree after DELETEMIN (no reasoning required): **2 points**
   (c) $x = 6$ **1 point**
   because of the heap property, it is bigger than 5 and smaller than 7 **2 points**
   $y = 4$ **1 point**
   because of the heap property, it is bigger than 3, but because it was used in downheapify instead of 5, it is smaller than 5 **2 points**

2. In a binary search tree containing all distinct integers, the order in which keys are encountered during a search for 11 is: $3, 17, 8, z, 11$. Determine all possible values of z.

   Grading guide:

   Correctly drawing the first three elements, 3's right child is 17 and 17's left child is 8, along with a reasoning (property of BSTs) **1+1 points**
   z is the right child, and its reasoning
   1+1 points
   Two cases for when 11 is the left or right child of z **1 point**
   The correct deductions for z because of the search path (greater than 3, smaller than 17 etc) **3 points**
   If 11 is the right child then $z = 9, 10$ **1 point**
   If 11 is the left child then $z = 12, 13, 14, 15, 16$ **1 point**
   Only a maximum of 1 out of the last 5 points can be awarded if there is no reasoning with the answer.
   If one case is missing (11 left/right child), then maximum of 8 points are awarded.

3. In an empty open-address hash table of size 11, we insert 7 distinct integers using the hash function $h(x) = x \pmod{11}$, with collisions resolved using a linear probe. After the insertion of all the 7 numbers, one of them is deleted, the position of the deleted element being marked with a *. The resulting hash table is given below:

0	1	2	3	4	5	6	7	8	9	10
		14	3		6	18	*	9	20	

(a) Show the steps taken in the search for 19 in this table. Indicate all the cells visited during this search.
(b) What can the deleted number be, if we know that it is a positive integer smaller than 20?

Grading guide:

If the solution demonstrates the knowledge that $h(k)$ is the starting search address **1 point**
If the solution demonstrates the knowledge that we decrease the address by 1 in case of collisions **1 point**
If the solution demonstrates the knowledge that we keep searching even after reaching a * (reasoning is not required) **2 points**
If the solution demonstrates the knowledge that we stop the search when we reach an empty cell (reasoning is not required) **1 point**
The hash value $h(x)$ of the deleted element can only be $7, 8, 9$ and reasoning **1+2 points**
With value 7 we can have 7 and 18, with 8 we can have 8 and 19, and with 9 we can only have 9. Of these 9 and 18 are there in the table and the numbers are distinct, so it can only be 7, 8 and 19. **2 point**
If the linear probe is done with increasing the address by 1 instead of decreasing it, then maximum of 2 and 3 points are awarded for the two subparts.

4. A large city currently has no bicycle lanes at all, but the new city government wants to see bicycle lanes painted on the outside lanes of some roads. Painting will be done on both sides of the selected roads, so that the bicycle lanes can be used in both directions. Unfortunately, converting the outer lanes to bicycle lanes will eliminate at least one parking space along the road. We are given the adjacency matrix of the simple, connected, undirected, weighted graph of the city, where vertices are nodes, edges are direct roads between the nodes, and the weight of an edge is the number of parking spaces eliminated if a bicycle lane is built along this road. The city administration would like to design the bicycle lanes so that it is possible to access all nodes from the main square (also a node), while eliminating as few few parking spaces as possible.
Give an algorithm with running time $O(n^2)$ for this problem (where n denotes the number of nodes).

Grading guide:

A Minimum spanning tree is required. **1 point**
Prim's algorithm is needed because of $O(n^2)$ running time. (Note: since I did not discuss this with clarity in recitations, points will not be deducted if Kruskal's is used.) **2 points**
Prim's can be started with any vertex as a spanning tree will connect everything to the main square. **1 point**
Running time is $O(n^2)$ because we are running Prim's once. **1 point**
We need to find a MST because to have a path from the main square to every node, the graph of bicycle roads must be connected. **2 points**
And since we are trying to minimize the cost, the graph will not have cycles, because then we can remove one bicycle road and get a graph with less weight. **2 points**
So we are looking for a Minimum weight spanning tree in this graph. **1 point**

5. Show that the following language is in NP:
42-CYCLE = $\{G|G$ is a simple, undirected graph such that its vertices can be covered with exactly 42 vertex disjoint cycles $\}$.
(So the cycles partition the vertex set of the graph, and every vertex of the graph is in exactly one cycle).

Grading guide:

A list of 42 cycles can be the witness. **1 point**
A graph has such a verifiable witness only if it can be partitioned into 42 cycles, and not otherwise. ($x \in L$ if and only if $(x, w) \in L'$ where l' is the witness language). **0 points**
with each cycle written as a list of vertices in the order they appear on the cycle. **1 point**
Size of input is $O(n^2)$ if using adjacency matrix, and $O(n^2 \log n)$ if using an adjacency list. **1 point**
Size of witness is $O(42n \log n)$ which is polynomial in the size of the input. **1 point**
Verification steps:

- Each listed cycle is indeed a cycle, that there are edges between adjacent vertices in the list. This can be done in polynomial time as it is just look up in adjacency matrix or list for each adjacent pair of vertices (total $O(n)$ such pairs). **1+1 point**
- Every vertex of the graph is listed in **exactly** one cycle. This can be checked in polynomial time by sorting the witness. **1+1 point**
- There are 42 cycles listed. We just go through the witness once to count the cycles, this is linear time. **1+1 point**

6. Show that 42-CYCLE as defined above is NP-complete. (You may use any result proved in the previous problem).

Grading guide:

From the previous solution, we know that this language is in NP. **1 point**
If we give a Karp reduction of an NP-complete language to this, then that would prove that this language is also NP-complete. **1 point**
HAMCYCLE is a good choice for the NP-complete language. **1 point**
Good reduction. Either 42 unconnected copies of G, or 41 new triangles not connected to each other or anything else. Basically any reduction which forces 41 cycles to be outside G. **2 points**
It is polynomial time to derive G' from G (increasing the adjacency matrix or adding to the adjacency list atmost linear number of new vertices). **1 point**
Proof that if G has a hamiltonian cycle then G' has a partition into 42 cycles. **2 points**
Proof that if G' has a partition into 42 cycles, then G has a Hamiltonian cycle. **2 points**

7. We want to record scores obtained by students in a multi-round competition, where students are identified by their Neptun code. Design a data structure that supports the following operations:
INSERT_STUDENT(x): insert a new student with Neptune code x in to the data structure with 0 points.
INCREASE_POINTS(x,y): the points of student with Neptun code x should be increased by y
POINTS(x): return the number of points of a student with Neptun code x
BEST_SCORE: return the Neptun code of all the students with the largest number of points.

We want the BEST_SCORE procedure to have a running time of $O(k)$, where k is the number of students with the most points. All the other operations must have a running time of $O(\log n)$, where n is the number of students participating in the competition.

Grading guide:

Use of RB-trees or 23-trees based on Neptun code.	**1 point**
An extra list or array for students with maximum points.	**2 points**
How the four operations are supported.	**1+1+1+1 points**
Running time of first three is $O(\log n)$ and last is $O(1)$.	**1+1 points**