IT3312E Data Structures & Algorithms, Spring 2021, Midterm exam

Name: ____________________ Student ID: ____________________ Signature: ____________________

Instructions

Assume your student id is 20195678. The least significant digit of this number is 8. The second least significant digit is 7. We have 4 integers x_0, x_1, x_2, x_3. Assign to x_0 = the least significant digit of your student id (here $x_0 = 8$). Assign to x_1 the second least significant digit of your student id (here $x_1 = 7$). Assign to x_2 the third least significant digit of your student id (here $x_2 = 6$). Assign to x_3 the fourth least significant digit of your student id (here $x_3 = 5$).

We have the following array A:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
A														

Fill the entries of A with the values as computed below and write your filled array A on your answer sheet. **Be very careful when you compute the values of A as errors could negatively impact your solutions to the midterm questions.**

- $A[0] = (x_0 \mod 6) + 1$; $A[1] = (x_1 \mod 7) + 1$; $A[2] = (x_2 \mod 8) + 1$; $A[3] = (x_3 \mod 9) + 1$;
- $A[4] = ((A[0] + A[2]) \mod 5) + 1$;
- $A[5] = ((A[1] + A[3]) \mod 5) + 1$;
- $A[6] = ((A[0] + A[1]) \mod 5) + 1$;
- $A[7] = ((A[0] + A[1] + A[2]) \mod 5) + 1$;
- $A[8] = ((A[1] + A[2]) \mod 5) + 1$;
- $A[9] = ((2A[0]) \mod 9) + 1$;
- $A[10] = ((4A[1]) \mod 8) + 1$;
- $A[11] = ((4A[2]) \mod 7) + 1$;
- $A[12] = ((3A[3]) \mod 7) + 1$;
- $A[13] = ((4A[4]) \mod 7) + 1$;

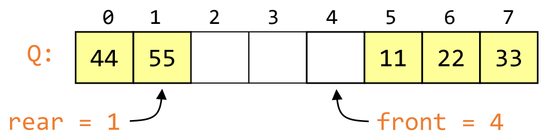
Question 1: Queues (18 pts)

Assume I declare the array A above as a 2 dimensional "C" array as follow: int $A[2][7]$. Let also assume assume that an "int" is 4 bytes long (don't forget, index values in C start at 0). (3 pts each)

1. List the elements of A as they will appear in the computer memory if the language stores the array A in row-major order
2. The array A is stored in row-major order starting at address 0 in the computer memory. What is the address of $A[1][3]$?
3. The array A is stored in row-major order starting at address 0 in the computer memory. Which value of A is stored at address 32?
4. List the elements of A as they will appear in the computer memory if the language stored the array A in column-major order
5. The array A is stored in column-major order starting at address 0 in the computer memory. What is the address of $A[1][3]$?
6. The array A is stored in column-major order starting at address 0 in the computer memory. Which value is stored at address 20?

Question 2: Queues (22 pts)

Consider the queue Q below:



Execute the sequence of operations on Q as dictated by the loop below. If an Enqueue() operation is attempted when the queue is full, skip such operation until the loop asks for a Dequeue() operation. If an Dequeue() operation is attempted when the queue is empty, skip such operation until the loop asks for an Enqueue() operation.

for $(i = 0;\ i < 8;\ i++)\{$
  if $(A[i] \leq A[i+1])$ Dequeue(Q);
  else Enqueue(Q,A[i]);
$\}$

1. Write on your answer sheet the value of each entry of Q after the execution of all the above operations. Note, each time Dequeue(Q) is performed, replace the corresponding entry in queue Q by -1.(16 pts)
2. What is the final value of the pointers *rear* and *front*? (6 pts)

Question 3: DP and 0-1 knapsack (23 pts)

You have a knapsack of capacity $W = 7$ and $n = 5$ objects where weights w_i (weight of object i) and values v_i (value of object i) are given in the table below:

i	1	2	3	4	5
w_i	$A[4]$	$A[5]$	$A[6]$	$A[7]$	$A[8]$
v_i	$A[9]$	$A[10]$	$A[11]$	$A[12]$	$A[13]$

1. Copy the table above on your answer sheet with the actual weights and values for your problem.
2. Copy the table below on your answer sheet. Fill the table using the dynamic programming algo for 0-1 knapsack and the actual weights and values of our problem instance above.(17 pts)

$i\backslash j$	0	1	2	3	4	5	6	7
1								
2								
3								
4								
5								

3. Based on your filled table, list the objects of the optimal solution (5 pts)

Question 4: Binary search tree (16 pts)

The binary search you will draw MUST satisfy the following property:

$$\mathbf{x.left.key < x.key \leq x.right.key}$$

1. Draw a binary search tree containing keys $A[0]$, $A[1]$, $A[2]$, $A[3]$, $A[4]$, $A[5]$, $A[6]$, $A[7]$, $A[8]$ inserted in this same order. (7 pts)
2. Redraw the binary search tree of the previous question after deleting the node with key $A[0]$ (the root node). (9 pts)

Question 5: Divide&conquer algorithms (25 pts)

Using the same 0-1 knapsack problem instance as in question 3, i.e. capacity $W = 7$ and $n = 5$ objects and weights and values as in this table.

i	1	2	3	4	5
w_i	$A[4]$	$A[5]$	$A[6]$	$A[7]$	$A[8]$
v_i	$A[9]$	$A[10]$	$A[11]$	$A[12]$	$A[13]$

Runs the following divide&Conquer algorithm on this instance

int K(i, W)
  if (i == 1) return (W < w[1]) ? 0 : v[1]
  if $(W < w[i])$ **return** K$(i-1, W)$;
  return max(K$(i-1, W)$, K$(i-1, W - w[i]) + v[i]$);

Output the results of running this algorithm using the full binary tree below (draw the tree on your answer sheet).

- On the left of each node write the subset of objects considered to build the solution returned for this node. For example, leaf nodes only consider object 1 ($i = 1$ in the table), so you write 1, while the root node considers all the objects, so you write 5 on the left side of the root node
- Inside each node you write the optimal solution returned by the recursive call corresponding to that node (subproblem)
- On the right of each node you write the capacity of the subproblem corresponding to the node. For example, for the root node you write 7 on the right side as this is the capacity for the whole problem instance

Only the solution of each subproblem (inside node) and capacity (on the right of each node) will be graded. There are 62 such values, to have full mark you need to have 50 of them correct. Each worths 0.5 pt.