A. Y. 2021 – 2022

FIRST AND SECOND SEMESTER

PRACTICE PREFINAL EXAM

CIS 2101 – DATA STRUCTURES AND ALGORITHMS

(ADT BINARY TREES, BINARY SEARCH TREES, PARTIALLY ORDERED TREES)

NOTES:

I. You have **150 minutes** to take the test. Kindly inspect the exam time duration carefully and do not spend too much time on a single question.

II. Each test has different directions. Follow them carefully.

III. Answer each item by typing your answers in the fields and spaces provided.

IV. Always try to write accurately using appropriate and efficient program logic and proper coding syntax. For programming problems, follow the coding conventional rules that are set for the class (e.g. all conditional statements must involve relational operators; break and continue statements shall be only used in switch blocks.)

V. If you are not certain with certain parts of your code, place your intended actions in the form of block comments within the area of concern for better analysis and feedback.

VI. Should you have a concern on any of the questions, kindly contact or message the examiner for inspection or clarification.

VII. Answer each question as far as you can. Try not to leave blanks.

VIII. Once you are finished with the exam, save the PDF file with your answers and submit it as an attachment to the examiner's email address (20100215@usc.edu.ph) or through direct message for checking.

Name:	**KATHLEEN IZA MONZALES**
Program and Year:	**BSCS - 2**
Date:	**MAY 08, 2022**

I. MULTIPLE CHOICE

DIRECTIONS: Choose the letter of the correct or best answer. [2 marks x 18 = 36 points]

E ..... 1. The level of a node in a binary tree is the distance from root to that node. For example, level of root is 0 and levels of left and right children of root is 1. The maximum number of nodes on level i of a binary tree is:
[A] 2^(i-1) [B] 2^(i+1) [C] 2^i [D] 2^[(i+1)/2] [E] None of the choices

C ..... 2. The height of a binary tree is the maximum number of edges in any root to leaf path. The maximum number of nodes in a binary tree of height h is:
[A] 2^h – 1 [B] 2^(h-1) – 1 [C] 2^(h+1) – 1 [D] 2^(h+1) [E] None of the choices

A ..... 3. In how many ways can a set with 3 elements be represented in a binary search tree?
[A] 3 [B] 4 [C] 5 [D] 1 [E] None of the choices

C ..... 4. Given an array of 12,345 distinct integers sorted in descending order, at most how many accesses to the array do you need in order to determine whether or not the target element exists using binary search operation?.
[A] 12 [B] 13 [C] 14 [D] 15 [E] None of the choices

E ..... 5. A scheme for storing binary trees in an array X is as follows. Indexing of X starts at 1 instead of 0. the root is stored at X[1]. For a node stored at X[i], the left child, if any, is stored in X[2i] and the right child, if any, in X[2i+1]. To be able to store any binary tree on N vertices the minimum size of X should be:

[A] $\log_2 N$ [B] N [C] 2N + 1 [D] 2^N – 1 [E] 2^N

B ..... 6. If the following elements {37, 46, 48, 40, 45, 60, 42} are inserted into an initially empty AVL tree, how many rotations are performed?
[A] 1 [B] 2 [C] 3 [D] 4 [E] None of the choices

B ..... 7. A programmer wants to construct a non-recursive function that traverses a given binary search tree in a preorder manner. What auxiliary data structure has to be accompanied?
[A] List [B] Stack [C] Queue [D} Both B and C [E] None of the choices

A ..... 8. Consider a node B in a Binary Tree. Given that B has two children, let C be the inorder successor of B. Which of the following is always true about C?
[A] C has no left child. [B] C has no right child.
[C] C has both left and right child. [D] C has no child.
[E] None of the choices.

D ..... 9. Which of the following is/are TRUE for any given binary search tree or subtree?
I. The largest member element does not contain a right child.
II. The largest member element does not contain a left child.
III. The largest member element may contain at most one child.
IV. The largest member element may not always be a leaf.

[A] I only [B] II only [C] III and IV [D] I, III, IV [E] II, III, IV [F] I and III [G] II and III

D ..... 10. A programmer wants to save the elements of a binary tree into an array such that when its members are reinserted through sequential accessing of the array, the order and structure can be maintained. In what order should the tree be traversed to meet his objective when saving the elements?

I. Preorder II. Postorder III. Inorder IV. Level order (Breadth first)

[A] I only [B] II only [C] III only [D] IV only [E] I and II [F] I and IV [G] III and IV

D ..... 11. The following operation/s in priority queue share the same running time when implemented via array and singly linked list (with pointers to first and last elements) where in both cases the elements are sorted in ascending order:
[A] insert() [B] deleteMin() [C] deleteMax() [D] B and C [E] A, B, and C [E] None of the choices

A ..... 12. If a heap is implemented to form a full binary tree with height 9 when the heap is fully populated and elements are populated starting at index 0, what must be the size of the array?
[A] 1023 [B] 1024 [C] 511 [D] 512 [E] None of the choices

C ..... 13. In a max-heap, the element with the greatest key is always in which node?
[A] Any leaf node [B] First node of left subtree [C] Root node [D] First node of right subtree
[E] Leftmost leaf [F] Rightmost leaf [G] None of the choices

A ..... 14. Consider a heap with 250 active elements and the root at index 0. If a heapify procedure is to be implemented, at what index of the array should the process start with?
[A] 249 [B] 0 [C] 125 [D] 124 [E] 123

C ..... 15. Which of the following sequence of integers represent a binary max heap?
[A] 25,12,16,13,10,8,14 [B] 25,12,16,13,10,8,14
[C] 25,14,16,13,10,8,12 [D] 25,14,12,13,10,8,16
[E] None of the choices

A ..... 16. A binary min heap, implemented using array, has 100 nodes. The root is stored at index 1. Which of the following statements is FALSE?
[A] The POT has at least 37 leaves.
[B] The POT has a maximum path of length 6.
[C] The element at index 1 must be less than or equal to that at index 2^n for all positive integers n.
[D] The last occupied element is at index 100.
[E] None of the choices

C ..... 17. A complete binary min-heap is made by including each integer in [1, 1023] exactly once. The depth of a node in the heap is the length of the path from the root of the heap to that node. Thus, the root is at depth 0. The maximum depth at which the integer 9 can appear is:
[A] 3 [B] 7 [C] 8 [D] 9 [E] None of the choices

B ..... 18. Which of the following is not a characteristic of a partially ordered tree?
[A] The height is always the minimum possible for the current number of nodes.
[B] At the lowest level, all the missing nodes are to the left of the nodes present at the lowest level.
[C] The priority of the parent is less than or equal to that of its children.
[D] Each node of the tree cannot exceed 2 immediate children.

II. STRUCTURED RESPONSE

DIRECTIONS: Read carefully and answer the questions correctly. To gain full marks to questions you should express your ideas sensibly and answer with the proper syntax. The number of marks is indicated in brackets [ ] at the end of each question or part question.

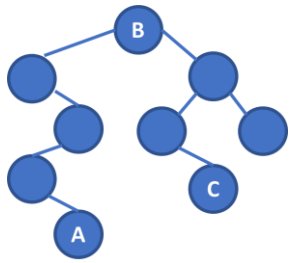
1. Complete the table below by filling in the worst possible time complexity to perform the following operations:

Operations	Binary Search Tree	AVL Tree	Priority Queue (Min Heap)	Priority Queue (Array w/ values sorted in descending order)
Inserting an element	O(h)	O(log2n)	O(log n)	O(log n)
Searching for an element	O(h)	O(log2n)	O(log n)	O(log n)
Searching for the minimum element	O(h)	O(log2n)	O(1)	O(n)
Deleting an element	O(h)	O(log2n)		
Deleting the minimum element	O(h-1)	O(log2n-1)	O(1)	O(1)
Displaying the elements in sorted order	O(n)		O(n)	

[10]

2. Values from 11 to 19 are assigned to 10 nodes in the binary search tree shown on the right. Determine the correct combination of values for A, B, and C.

(Note: Points will be awarded only when all answers are correct)



A	B	C
13	15	17

[4]

3. A binary tree is implemented using a **list of children representation**. See the values below:

Parent	0	1	2	3	4	5	6	7	8	9
Left Child	-1	9	-1	2	-1	3	4	-1	-1	-1
Right Child	-1	6	7	0	8	1	-1	-1	-1	-1

Note: -1 means no child

(a) Determine the height of the tree. **4** **[1]**

(b) Determine the depth of node 0. **2** **[1]**

(c) How many paths of length 2 are there? **7** **[2]**

(d) How many paths of length 3 are there? **3** **[2]**

(e) Identify the leaves of the tree. **4** **[2]**

(f) Determine the ancestors of node 0. **3, 5** **[2]**

(g) List the nodes of the longest path. **5, 1, 6, 4, 8** **[1]**

(h) What is the lowest level node that is not height-balanced? **6** **[1]**

(i) Provide the inorder listing of nodes. **2, 7, 3, 0, 5, 9, 1, 4, 8, 6** **[3]**

(j) Use the code on the right to simulate the traversal of the tree, where datatype BST is a pointer to a node containing its label and pointers to left and right child nodes. How many times is traverseBST() called until the tree traversal terminates? **16** **[3]**

```
void traverseBST(BST A){
   if(A!=NULL){
      printf("%c",A->elem);
      traverseBST(A->leftChild);
      traverseBST(A->rightChild);
   }
}
```

(k) Elements 3, 5, and 7 are to be deleted in the binary tree. If a node to be deleted has two children, its **inorder succesor** will be replaced following appropriate adjustments in the tree. Provide the preorder listing of nodes after the deletion of the three elements. **2, 0, 8, 4, 6, 1, 9** **[4]**

4. Given: **int A[]={1,3,6,10,15,21,24,30,34,38,40,46,50,58,65,68}**

(a) List the possible elements that may be accessed during the second iteration of the binary searching of the array. **10, 46** **[2]**

(b) Using binary search algorithm, how many accesses are needed to search for the element containing the number 68? **5** **[3]**

(c) The values above will be inserted into a binary search tree starting from the first to the last element. Explain the effects of this implementation.

It will not satisfy the properties of a binary search as the elements are inserted in an ascending order, the left child will be greater than their parent.

[3]

5. A programmer is writing the code of the function which removes and returns the information contained in the node, given a binary search tree and the target element to be searched. He argues that the binary search tree can be passed to the function through a pointer to the root node because the root will always contain two children, and when deleting a node (like the root) with two children, the node will not be removed but rather replaced by either its inorder predecessor or successor. Is his action and claims correct? Support your answer with **accurate reasons** and provide resolutions should his claim be flawed.

Passing a pointer that points to the root node to a function is correct because it is the efficient way to manipulate a binary search tree (inserting/deleting an element), however, a root node does not always have two children. At times, a root node can only be the only node in the tree, at such it has no children at all, or it may only contain one child, either left or right child. With this, his code in the function has to have 2 conditions, one for when the node has only one child or no child, and another one for when the node has 2 children wherein the function finds the inorder predecessor or successor to replace the node to be removed from the tree.

[5]

6. Insert the following values **{4,6,10,9,5,2,1,0,16,18}** into an initially empty AVL tree.

(a) Identify the height of the tree. **3** **[1]**

(b) Provide the preorder traversal of the resulting AVL tree. **2, 0, 1, 6, 5, 18, 10, 16, 9, 4** **[5]**

(c) Determine the number of: (i) Single rotations: **1** **[2]**

(ii) Double rotations: **2** **[2]**

7. (a) A max heap is implemented in an array whose root is located at index 0.

(i) The right child of index 26 is at index **54** **[1]**

(ii) The left child of index 37 is at index **75** **[1]**

(iii) The parent of index 74 is at index **36** **[1]**

(iv) At most how many swaps within the heap will occur if a new element is inserted at index 42 and the rest of the elements before that are occupied and form a heap? **4** **[2]**

(v) Within indices 41-60 of the heap, the element at index 5 must be always larger than or equal to the elements in what indices? **47, 48, 49, 50, 51, 52, 53, 54** **[3]**

(b) A min heap is implemented in an array whose root is located at index 1.

(i) The right child of index 9 is at index **19** **[1]**

(ii) The left child of index 12 is at index **24** **[1]**

(iii) The ancestors of index 30 excluding itself are at indices **15, 7, 3, 1** **[3]**

(iv) If a new element is initially inserted at index 67, the possible final positions of this element are at indices **67, 33, 16, 8, 4, 2, 1** **[3]**

8. A programmer implements a priority queue using a regular array implementation (not a heap). He implements it that way as he claims that the deleteMin() operation can be done in constant (O(1)) time.

(a) Is this scenario attainable? When is it so? **if the order of the elements when inserted to the array is by ascending order** **[2]**

(b) Cite the effects of this implementation on the other operations related to priority queues.

If ascending order, it would take O(n) when we insert a new node and attempt to keep the ascending order of the array, because we would be swapping the new element with the element before them instead of evaluating their parent node when heap is used.

[3]

9. Christine, a DSA enthusiast, likes visiting Japan. So in order for her trip there to go well, she spends time learning the Japanese Language by studying few vocabulary terms every day. Below shows the new words she got today:
sensei, kirei, namae, konpiuta, kyou, sumimasen, arigato, doitashimashite, konbanwa, wakarimasen

(a) Using an array of strings *arr1*, build the **min-heap** by inserting the given strings in order into an initially empty POT. Complete the table by identifying the strings present in the selected array indices.

Scenario	A[0]	A[3]	A[5]
Initial heap	arigato	kirei	sumimasen
After removing two (2) smallest elements	namae	sumimasen	konpiuta

[6]

(b) Assuming the given strings appear in the exact order in an array of strings *arr2*, build the **max-heap** by heapifying starting from the lowest level parent. Complete the table by identifying the strings present in the selected array indices.

Scenario	A[0]	A[3]	A[5]
Initial heap	wakarimasen	kirei	sensei
After removing two (2) largest elements	kyou	sensei	doitashimashite

[6]

III. PROGRAMMING

DIRECTIONS: Read carefully and give accurately what is asked. Make your code concise, efficient, and readable. Follow the coding conventions that are set for the class. The number of marks is indicated in brackets [ ] at the end of each problem or subproblem.

1. Given an array of integers (with -1 at the end which is not part of the list but rather signifies the end of the list):

(a) Write the code of function **isMaxHeap()**. The function checks if the given array represents a max heap and returns 1 if so and 0 if it isn't. **[10]**

(b) Write the code of function **buildHeap()**. The function builds a min heap by first copying the elements directly to a new array then applies the heapify procedure starting from the lowest level parent. The resulting array will then be returned to the calling function. **[15]**

2. ABC Restaurant has a program which tracks the inventory of their menu items using an internal binary search tree, organized by item_ID. See the data structure definition on the right.

```
typedef struct product{
   char prod_ID[10];
   char prod_name[50];
   int qty_remaining;
}prodType; /*Item record*/
```

```
typedef struct node{
   prodType prod;
   struct node *LChild, *RChild;
}nodeType, *BST;
```

Implement the following functions/prototypes:

(a) **BST * findMemberPos(BST *B, char prod_ID[]);** - the non-recursive function will return a pointer to a pointer to the node containing the given item ID. If the item is not found, the pointer returned points a pointer variable with a value of NULL. **[7]**

(b) **Function deleteMinProd()**. The non-recursive function deletes the node holding the product record with the smallest product ID and returns the item record contained in the node. **[7]**

(c) **Function decreaseInventory().** Given the BST and the item record, the function searches for the product using findMemberPos(). If the record is found, the function decreases the item quantity of the record in BST by the quantity of the item given as parameter in the function, if the item quantity in the BST is larger than the latter quantity. Otherwise, the item record shall be removed from the inventory. Invoke function deleteMinProd() whenever necessary. **[16]**

```
/* For question (1a) */ - assuming root node is at index 0

int isMaxHeap(int heap[]){
   int isMaxHeap = 1, x;

   /* traverse array until left child is -1 */
   for(x = 0; heap[x]*2+1 != -1; x++){

      /* if either left or right child is smaller than parent */
      if(heap[x]*2+1 < heap[x] || heap[x]*2+2 < heap[x]){
         isMaxHeap = 0;
         break;
      }
   }
   return isMaxHeap;
}
```

```
/* For question (1b) */ - assuming root node is at index 0

int* buildHeap(int heap[]){
   int arr[sizeof(heap)/sizeof(int)];
   int x, ctr;

   /* copy the elements of the heap[] to the arr[] */
   for(x = 0; heap[x] != -1; x++){
      arr[x] = heap[x];
   }
   arr[x] = -1;

   /* perform min-heapify starting from lowest level parent */
   for(ctr = x-1; ctr > 0; ctr--){
      int parent = ctr/2;
      /* swap if parent is greater than their parent */
      if(arr[ctr] > arr[parent]){
         int temp;
         temp = arr[ctr];
         arr[ctr] = arr[parent];
         arr[parent] = temp;
      }
   }
   return arr;
}
```

```
/* For question (2a) */

BST *findMemberPos(BST *B, char prod_ID[]){

   /* BST *B points to a pointer that points to the head of the BST*/
   while(strcmp((*B)->prod.prod_ID, prod_ID) != 0 && *B != NULL){
      B = strcmp((*B)->prod.prod_ID, prod_ID) > 0 ? &(*B)->LChild : &(*B)->RChild;
   }
   return B;
}


/* For question (2b) */

prodType deleteMinProd(BST *B){
   prodType product;
   BST ptr;

   /* traverse to the smallest product ID */
   for(; (*B)->LChild != NULL; B = &(*B)->LChild){}
   product = (*B)->prod;
   ptr = *B;
   *B = ptr->RChild;
   free(ptr);
   return product;
}


/* For question (2c) */

void decreaseInventory(BST *B, char prod_ID[], int decrease){
   /* find the position of the product */
   B = findMemberPos(B, prod_ID);
   BST ptr;
   prodType temp;
   if(*B != NULL){
      /* if the qty remaining is smaller than the decrease, remove the node*/
      if(decrease > (*B)->prod.qty_remaining){
         ptr = *B;
         /* if the node has 2 children */
         if((*B)->LChild != NULL && (*B)->RChild != NULL){
            temp = deleteMinProd(&(*B)->RChild);
            ptr->prod = temp;
         } else {
            *B = (ptr->LChild != NULL)? ptr->LChild : ptr->RChild;
            free(ptr);
         }
      } else {
         (*B)->prod.qty_remaining -= decrease;
      }
   }
}
```

"If you're always trying to be normal, you will never know how amazing you can be."
- Maya Angelou -

=== THE END ===

God bless you!

REVIEW YOUR ANSWERS!

W. Dayata