BINUS University

Academic Career: ***Undergraduate / ~~Master / Doctoral~~ *)***			**Class Program:** ***~~International~~/Regular/~~Smart Program~~/Global Class*)***
☑ **Mid Exam** ☐ **Short Term Exam**	☐ **Final Exam** ☐ **Others Exam : _____**		**Term : ~~Odd~~/Even/~~Short~~ *)**
☑ **Kemanggisan** ☐ **Senayan**	☑ **Alam Sutera** ☐ **Bandung**	☐ **Bekasi** ☐ **Malang**	**Academic Year :** **2020 / 2021**
Faculty / Dept. : School of Computer Science		Deadline	Day / Date : Tuesday / Apr 20th, 2021 Time : 13.00 – 16.20 (200 Minutes)
Code - Course : COMP6048 – Data Structures		Class : All Classes	
Lecturer : Team		Exam Type : Online	
*) *Strikethrough the unnecessary items*			
The penalty for CHEATING is DROP OUT!!!			

EXAM INSTRUCTIONS

1. For Essay (Number 1 and 2)
- **A.** If your **Student ID (NIM)** is odd then please solve **1 a & c and 2 a**
- **B.** If your **Student ID (NIM)** is even then please solve **1 b & d and 2 b**

2. Convert your essay answers (1,2,3) in 1 PDF file using this format: NIM.pdf

3. For case problem: a. The submission code is in .cpp file and using this format: NIM.cpp

4. All your answers either essay (NIM.pdf) or case (NIM.cpp) should be zipped and submitted through https://exam.apps.binus.ac.id/. Other than that, the submission won't be accepted for any reasons. (Note : Please zip both files using this format: NIM.zip)

6. It is strictly forbidden to share answers with other students, or to take any reference sources from anywhere. Answers that are indicated as fraud will be reported!

7.The exam will be marked as 0 if any plagiarism is found, or you solve A WRONG PROBLEM SET

Learning Outcomes:

LO1 : Explain the concept of data structures and its usage in Computer Science
LO2 : Illustrate any learned data structure and its usage in application
LO3 : Apply data structures using C

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021

I. Essay (55%)

1. Applications of Stack (LO1, LO2 - 20%)

a. **Convert (A+(B*(C-(D/E))))-F** to **Postfix** Notation. Simulate it using **Stack** Algorithm
b. **Convert (A-(B*(C+D))-E/F)** to **Prefix** Notation. Simulate it using **Stack** Algorithm
c. **Evaluate** Postfix **9 10 5 – 7 + 2 /+** . Simulate it using **Stack** Algorithm
d. **Evaluate** Infix **+ - * / 14 7 3 4 / 9 3** . Simulate it using **Stack** Algorithm

2. Hashing & Collision (LO1, LO2 - 15%)

a. Using **Division Method** then **Linear Probing** if collision happens, insert keys **51, 22, 33, 45, 81, 60, 63, 69, 60, 55, 21** into hash table of size **11**
b. Calculate the hash value using **Folding Method** for keys **1921678, 777281, 1892, 1921, 21017** for hash table with **1000** memory locations

3. Snippet (LO1, LO2, LO3 - 20%)

Given snippet code below that you are required to complete. You are not allowed to make a new function or change any given code. Please complete each section that are marked with the notation "**INSERT YOUR CODE HERE**". Once you complete the snippet below, your output should have the same result with the given output below.

Description:

a. search()
   This function is built for searching whether the data has the same value as the node in the existing Binary Search Tree. If it is found then print "found", otherwise print "not found"
b. insert()
   This function is built for inserting a new node in the existing Binary Search Tree
c. inorder_traversal ()
   This function is built for printing out the existing Binary Search Tree in inorder way.
d. preorder_traversal ()
   This function is built for printing out the existing Binary Search Tree in preorder way.
e. postorder_traversal ()
   This function is built for printing out the existing Binary Search Tree in postorder way.
f. executeDeleteNode ()
   This function is built for deleting a node in the existing Binary Search Tree. If the node is either root or parent, then find replacement for the node otherwise delete the leaf node.
g. deleteNode()
   This function is built for searching the node in the existing Binary Search Tree that want to be deleted
h. minValue()
   This function is built for searching the minimum value in the existing Binary Search Tree.
i. maxValue()
   This function is built for searching the maximum value in the existing Binary Search Tree.

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021

j. count()
This function is built for counting how many nodes in the existing Binary Search Tree

k. height_of_binary_tree()
This function is built for knowing the height of the existing Binary Search Tree

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>

struct node {
   int key;
   struct node *left;
   struct node *right;
} *root;

void search(struct node* curr, int find) {
   //INSERT YOUR CODE HERE
}

void insert(struct node* curr, int val) {
   struct node *new_node = (struct node*) malloc(sizeof(struct node));
   new_node->key = val;
   new_node->left = NULL;
   new_node->right = NULL;

   if (root == NULL) {
         root = new_node;
   }
   else if (val != curr->key) {
         if (val < curr->key && curr->left == NULL) {
               curr->left = new_node;
         }
         else if (val > curr->key && curr->right == NULL) {
               curr->right = new_node;
         }
         else if (val < curr->key && curr->left != NULL) {
               insert(curr->left, val);
         }
         else if (val > curr->key && curr->right != NULL) {
               insert(curr->right, val);
         }
   }
}
void inorder_traversal(struct node* curr) {
   if (curr == NULL) return;
   inorder_traversal(curr->left);
   printf("%d ", curr->key);
   inorder_traversal(curr->right);
}

void preorder_traversal(struct node* curr) {
   //INSERT YOUR CODE HERE
}

void postorder_traversal(struct node* curr) {
   //INSERT YOUR CODE HERE
}

void executeDeleteNode(struct node* parent, struct node* curr) {
   //INSERT YOUR CODE HERE
```

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021

```
}

void deleteNode(struct node* curr, int find) {
   struct node *parent;
   parent = curr;
   while (curr != NULL && curr->key != find) {
         if (find < curr->key) {
               parent = curr;
               curr = curr->left;
         }
         else if (find > curr->key) {
               parent = curr;
               curr = curr->right;
         }
   }
   if (curr == NULL) {
         printf("%d is not found");
   }
   else if (curr->key == find) {
         executeDeleteNode(parent, curr);
   }
}

int minValue(struct node* node) {
   //INSERT YOUR CODE HERE
}

int maxValue(struct node* node){
   //INSERT YOUR CODE HERE
}

int count(struct node* node){
   //INSERT YOUR CODE HERE
}

int height_of_binary_tree(struct node *node){
   //INSERT YOUR CODE HERE
}

int main() {
   root = NULL;
   insert(root, 54);
   insert(root, 81);
   insert(root, 16);
   insert(root, 27);
   insert(root, 9);
   insert(root, 36);
   insert(root, 63);
   insert(root, 72);

   printf("Total node in BST: %d \n",count(root));
   printf("Height of binary tree : %d\n ", height_of_binary_tree(root));

   printf("\nInorder: "); inorder_traversal(root);
   printf("\nPreorder: "); preorder_traversal(root);
   printf("\nPostorder: "); postorder_traversal(root);

   puts("");
   deleteNode(root, 54);
   printf("\nPreorder (after del 54): "); preorder_traversal(root);
```

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021

```
    deleteNode(root, 9);
    printf("\nPostorder (after del 9): "); postorder_traversal(root);
    deleteNode(root, 81);
    printf("\nInorder (after del 81): "); inorder_traversal(root);

    puts("");
    printf("Min value: %d \n", minValue(root));
    printf("Max value: %d \n", maxValue(root));

    puts("");
    search(root, 10);
    search(root, 8);

    _getch();
    return 0;
}
```

Output:

```
Total node in BST: 8
Height of binary tree : 4

Inorder: 9 16 27 36 54 63 72 81
Preorder: 54 16 9 27 36 81 63 72
Postorder: 9 36 27 16 72 63 81 54

Preorder (after del 54): 36 16 9 27 81 63 72
Postorder (after del 9): 27 16 72 63 81 36
Inorder (after del 81): 16 27 36 63 72
Min value: 16
Max value: 72

72 is found
8 is not found
```

II. Case (LO1, LO2, LO3 - 45%)

Hospital XX has a program to manage patient queues with the following criteria:

1. Queue priority is set based on the patient's condition in the following order: Critical > Serious > Fair > Good
2. If the patient has the same condition then First In First Served
3. There are two types of events: ADD and CALL
4. ADD to insert patient to the queue
5. CALL to move the patient into the room based on condition:
   a. If patient in "Critical" Condition then move to Emergency Room
   b. If patient in "Serious" Condition then move to the Examination Room
   c. If patient in "Fair" or "Good" Condition move to the Consultation Room

Try implementing Double Linked List into the Hospital XX program using C

FORMAT INPUT

The first line contains an integer N events. Each event contains three variables String C1, String C2, String C3. String C1 with format "C1 C2 C3"

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021

FORMAT OUTPUT

If "CALL" is triggered then

1. If patient in "Critical" Condition then the output format will be : "C2 is in the Emergency Room"
2. If patient in "Serious" Condition then the output format will be : "C2 is in the Examination Room"
3. If patient in "Fair" or "Good" Condition then the output format will be : "C2 is in the Consultation Room"

Print the remaining patients in Waiting Room. If there is no patient then print None

CONSTRAINTS

1≤N≤100

C1 will contain "**ADD**" or "**CALL**"

3≤|C2|≤30

C3 will contain "**Good**","**Fair**","**Serious**" or "**Critical**"

Sample 1:

```
9
ADD Roddy Good
ADD Celine Fair
ADD Berry Good
CALL
Celine is in the Consultation Room
ADD Ronny Critical
ADD Belany Serious
CALL
Ronny is in the Emergency Room
CALL
Belany is in the Examination Room
CALL
Roddy is in the Consultation Room
Waiting Room: Berry

--------------------------------
Process exited after 0.05033 seconds with return value 0
Press any key to continue . . .
```

Sample 2:

```
6
ADD Roddy Good
ADD Celine Fair
ADD Ronny Critical
ADD Belany Serious
CALL
Ronny is in the Emergency Room
CALL
Belany is in the Examination Room
Waiting Room: Celine, Berry

--------------------------------
Process exited after 0.04427 seconds with return value 0
Press any key to continue . . .
```

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021

Sample 3:

```
8
ADD Roddy Good
ADD Celine Fair
CALL
Celine is in the Consultation Room
ADD Ronny Critical
ADD Belany Serious
CALL
Ronny is in the Emergency Room
CALL
Belany is in the Examination Room
CALL
Roddy is in the Consultation Room
Waiting Room: None

--------------------------------
Process exited after 0.05276 seconds with return value 0
Press any key to continue . . .
```

-- Good Luck --

Verified by,

Ajeng Wulandari (D6422) and sent to Program on March 30, 2021