BINUS University

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Academic Career:**  ***Undergraduate / ~~Master~~ / ~~Doctoral~~ \*)*** | | | | | | | | | **Class Program:**  ***~~International~~ / Regular / ~~Smart Program~~ /***  ***~~Global Class~~ / ~~BINUS Online Learning~~ \*)*** | | |
| **🗹 Mid Exam**  **🞏 Final Exam** | | | | **🞏 Compact Term Exam**  **🞏 Others Exam : \_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | **Term : ~~Odd~~ / Even** / **~~Compact~~ \*)**  **Period (Only for BOL) : ~~1~~ / ~~2~~ \*)** | | |
| **🗹 Kemanggisan**  **🗹 Alam Sutera**  **🗹 Bekasi** | | | **🞏 Senayan**  **🞏 Bandung**  **🞏 Malang** | | | | **🞏 Semarang** | | **Academic Year :**  **2022 / 2023** | | |
| Exam Type\* | | : | | | Onsite / ~~Online~~ | | | | Faculty / Dept. | : | School of Computer Science |
| Day / Date\*\* | | : | | | Thursday / April 13rd 2023 | | | | Code - Course | : | COMP6048 - Data Structures COMP6048001 - Data Structures  COMP6048016 - Data Structures  COMP6048049 - Data Structures |
| Time\*\* | | : | | | 08:00 – 11:20 (200 minutes) | | | | Code - Lecturer | : | Team Teaching |
| Exam Specification\*\*\* | | : | | | **🞏** Open Book  **🗹** Close Book  **🞏** Open E-Book | | | **🞏** Open Notes  **🞏** Submit Project  **🞏** Oral Test | BULC (Only for BOL) | : |  |
| Class | : |  |
| Equipment\*\*\* | | : | | |  | | | | Student ID \*\*\* | : |  |
| **🞏** Exam Booklet **🞏** Calculator **🞏** Dictionary | **🞏** Laptop  **🞏** Tablet  **🞏** Smartphone | | | | | **🞏** Drawing Paper – A3  **🞏** Drawing Paper – A2  **🞏** Notes | | | Name \*\*\* | : |  |
| Signature \*\*\* | : |  |
| 🞸) *Strikethrough the unnecessary items \*\*) For Online Exam, this is the due date* \*\*\*) *Only for Onsite Exam* | | | | | | | | | | | |
| Please insert the test paper into the exam booklet and submit both papers after the test.  ***The penalty for CHEATING is DROP OUT!*** | | | | | | | | | | | |

***Learning Outcomes:***

LO 1 : Explain the concept of data structures and its usage in Computer Science

LO 2 : Illustrate any learned data structures and its usage in application

LO 3 : Apply data structures using C

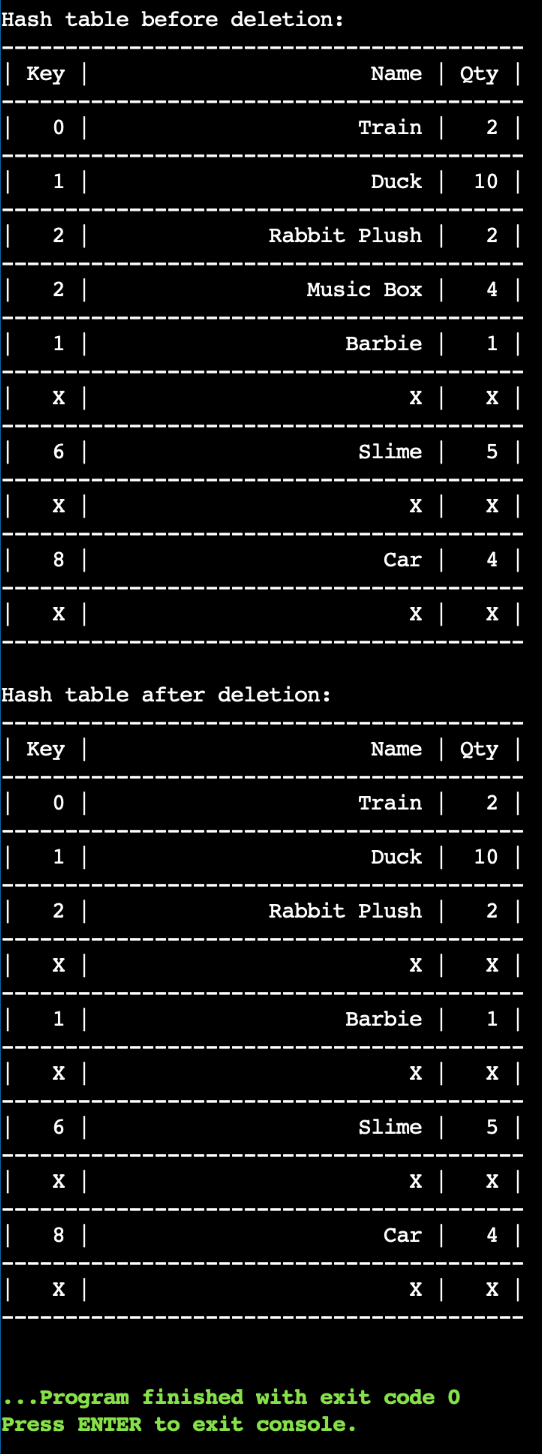
1. Essay (60%)
   * + 1. Applications of Stack
2. [LO 1, LO 2, 10 points] Convert (A\*(B+(C\*D)^(E/F))) to Postfix Notation. Simulate it using Stack Algorithm
3. [LO 1, LO 2, 10 points] Evaluate prefix + \* / \* 4 5 - + 6 / ^ 2 4 8 3 6 1. Simulate it using Stack Algorithm

\*Answer the question on Microsoft Excel. The file extension to be uploaded is .xlsx

* + - 1. [LO 1, LO 2 & LO 3, 20 points] Complete the hashing code below using Division & Linear Probing function!

|  |
| --- |
| #include<stdio.h>  #include<string.h>  #include<stdlib.h>  #define SIZE 10  struct Toy{  char name[20];  int quantity;  }\*box[SIZE];  int count = 0;  struct Toy\* new\_toy(char name[], int quantity){    **// Complete this part of code to create new toy called myToy (2 points)**  return myToy;  }  int hashing(char name[])  {  **/\*Complete this part of code with Looping to hash the name of toy (6 points)**  Example: Car  C  Key: 67  a  Key : 67 + 97 : 164  r  Key: 164 + 114 : 278  Hashing: 278 % 20 = 18  \*/  return key%SIZE;  }  void insert(char name[], int quantity){    int hashkey = hashing(name);  **// Complete the code to insert new data to hash table (6 points)**  **// if the box full show "The Toy Box is Full" and don’t add the *count* variable**  **// if there's any collision, solve it using linear probing**  count++;  }  void view(){  int i = 0;  printf("-----------------------------------------\n");  printf("| %3s | %25s | %3s |\n", "Key", "Name", "Qty");  printf("-----------------------------------------\n");  for(int i = 0; i < SIZE; i++){  if(box[i]){  printf("| %3d | %25s | %3d |\n", hashing(box[i]->name), box[i]->name, box[i]->quantity);  printf("-----------------------------------------\n");  } else {  printf("| %3s | %25s | %3s |\n", "X", "X", "X");  printf("-----------------------------------------\n");  }  }  }  void del(char name[])  {  int hashkey= hashing(name);  int point= hashkey;    do{  **//Complete this part of code to delete the toy (6 points)**  }while(point != hashkey);    return;  }  int main(){  insert("Duck",10);  insert("Train",2);  insert("Rabbit Plush", 2);  insert("Music Box",4);  insert("Slime",5);  insert("Barbie",1);  insert("Car",4);  printf("Hash table before deletion:\n");  view();  del("Music Box");  printf("\n");  printf("Hash table before deletion:\n");  view();    return 0;  } |

Output:



\*Answer the snippet on Dev C++. The file extension to be uploaded is .cpp

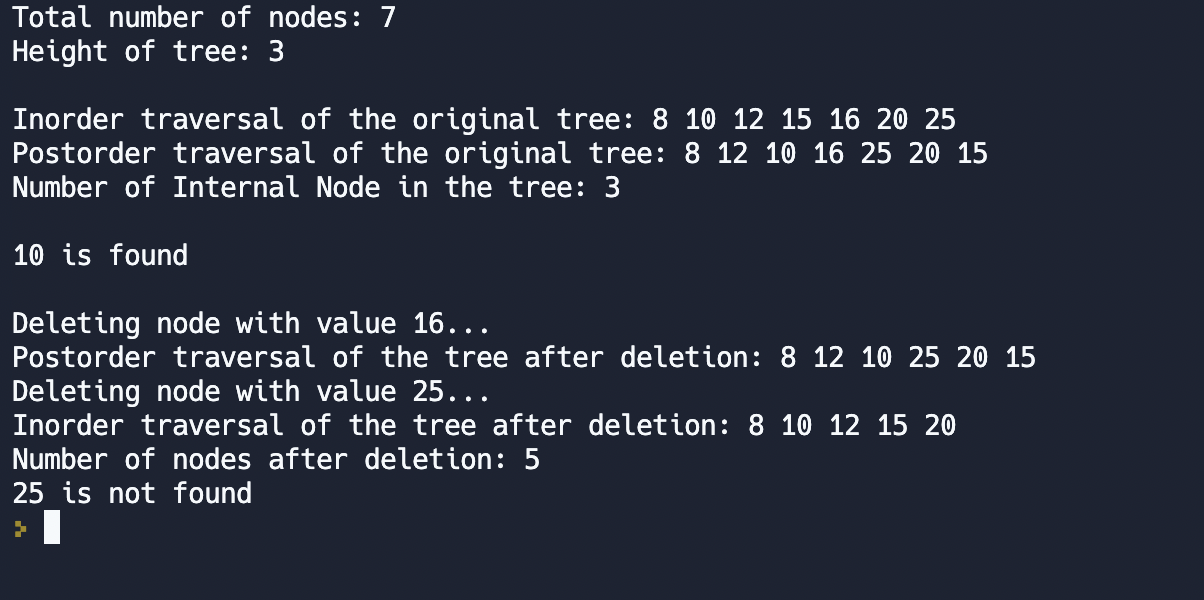
* + - 1. [LO 1, LO 2 & LO 3, 20 points] Given snippet code below that you are required to complete. You are not allowed to make a new function or change any given code. Please complete each section that are marked with the notation “INSERT YOUR CODE HERE”. Once you complete the snippet below, your output should have the same result with the given output below.

Function description :

1. **newNode(),** this function to create a new node
2. **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”
3. **insert(), t**his function is built for inserting a new node in the existing Binary Search Tree
4. **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.
5. **deleteNode()**, this function is built for searching the node in the existing Binary Search Tree that want to be deleted
6. **countNode()**, this function is built for counting how many nodes in the existing Binary Search Tree.
7. **countInternalNodes(),** this function is built for counting how many internal node in the existing Binary Search Tree. Internal node is a non-leaf node (node of a tree that has one or more child nodes).
8. **findHeight()**, this function is built for knowing the height of the existing Binary Search Tree
9. **inorderTraversal ()**, this function is built for printing out the existing Binary Search Tree in inorder way.
10. **postorderTraversal (),** this function is built for printing out the existing Binary Search Tree in postorder way.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  // define the binary search tree node structure  struct Node {  int data;  struct Node \*left;  struct Node \*right;  };  struct Node \*newNode(int data) {  **//INSERT YOUR CODE HERE (2 points)**  }  // Insert node  struct Node \*insert(struct Node \*node, int data) {  if (node == NULL) {  return newNode(data);  }  if (data < node->data) {  node->left = insert(node->left, data);  } else if (data > node->data) {  node->right = insert(node->right, data);  }  return node;  }  void search(struct Node \*current, int find) {  **//INSERT YOUR CODE HERE (5 points)**  }  void executeDeleteNode(struct Node \*node, struct Node \*current) {  if (current->left == NULL && current->right == NULL) {  if (current == node->left) {  node->left = NULL;  free(current);  } else if (current == node->right) {  node->right = NULL;  free(current);  }  } else if (current->left != NULL && current->right == NULL) {  if (current == node->left) {  node->left = current->left;  free(current);  } else if (current == node->right) {  node->right = current->left;  free(current);  }  } else if (current->left == NULL && current->right != NULL) {  if (current == node->left) {  node->left = current->right;  free(current);  } else if (current == node->right) {  node->right = current->right;  free(current);  }  } else if (current->left != NULL && current->right != NULL) {  struct Node \*p = NULL;  struct Node \*pp = NULL;  pp = current;  p = current->left;  while (p->right != NULL) {  pp = p;  p = p->right;  }  current->data = p->data;  executeDeleteNode(pp, p);  }  }  void deleteNode(struct Node \*current, int find) {  struct Node \*node;  node = current;  while (current != NULL && current->data != find) {  if (find < current->data) {  node = current;  current = current->left;  } else if (find > current->data) {  node = current;  current = current->right;  }  }  if (current == NULL) {  printf("%d is not found \n", find);  } else if (current->data == find) {  executeDeleteNode(node, current);  }  }  int countNodes(struct Node \*root) {  if (root == NULL) {  return 0;  }  return 1 + countNodes(root->left) + countNodes(root->right);  }  int findHeight(struct Node \*node) {  **//INSERT YOUR CODE HERE (2 points)**  }  int countInternalNodes(struct Node \*node) {  **//INSERT YOUR CODE HERE (3 points)**  }  void inorderTraversal(struct Node \*node) {  **//INSERT YOUR CODE HERE (4 points)**  }  void postorderTraversal(struct Node \*node) {  **//INSERT YOUR CODE HERE (4 points)**  }  int main() {  struct Node \*root = NULL;  // insert some nodes into the BST  root = insert(root, 15);  insert(root, 10);  insert(root, 8);  insert(root, 12);  insert(root, 20);  insert(root, 16);  insert(root, 25);  printf("Total number of nodes: %d\n", countNodes(root));    printf("Height of tree: %d\n", findHeight(root));  puts("");  printf("Inorder traversal of the original tree: ");  inorderTraversal(root);  printf("\n");  printf("Postorder traversal of the original tree: ");  postorderTraversal(root);  puts("");    printf("Number of Internal Node in the tree: %d\n", countInternalNodes(root));  puts("");  search(root, 10);  puts("");  printf("Deleting node with value 16...\n");  deleteNode(root, 16);  printf("Postorder traversal of the tree after deletion: ");  postorderTraversal(root);  puts("");  printf("Deleting node with value 25...\n");  deleteNode(root, 25);  printf("Inorder traversal of the tree after deletion: ");  inorderTraversal(root);  puts("");  printf("Number of nodes after deletion: %d\n", countNodes(root));  search(root, 25);  return 0;  } |

Output:



\*Answer the snippet on Dev C++. The file extension to be uploaded is .cpp

1. Case (40%)

[LO 3, 40 points] Coco owns a theme park called "Istana Boneka" which uses boats to explore the ethnic cultures throughout Indonesia. Each boat can accommodate 4 visitors. Coco sells 2 types of tickets, namely FAST\_TRACK and REGULER. The FAST\_TRACK ticket allows visitors to skip the REGULER queue and get priority. If visitors have the same type of ticket, the first one to register will be given priority (First In First Serve). To make it easier, Coco wants to create a queue program using the priority queue concept with a double linked list. The program has 3 main features:

REGISTER, used to add visitors. The REGISTER feature has the format REGISTER N1, where N1 is the number of visitors who want to play. In the next line, program will ask you to enter name visitor and ticket types according to the number of N1. After successfully adding, the queue list will be displayed in the format "No Name Ticket".

CALL, used to call the top 4 visitors in the queue. After successfully called, the list of visitors who board the boat will be show in the format "[list of visitors] got into the boat" and the remaining queue in the format "[total remaining queue] queues remaining". If there are less than 4 visitors in the queue, only those visitors in the queue will be called.

REMOVE, used to remove visitors who did not show up. The REMOVE feature has the format REMOVE N2, where N2 is the number of visitors who did not show up. In the next line, program will ask you enter names of visitors who did not show up according to the number of N2. After successfully removing, the queue list will be displayed in the format "No Name Ticket".

Contraints:

1 <= N1 <=1000

1 <= N2 <=1000

1 <= |name of visitor| <=10

1 <= |ticket types| <=10

|name of visitor| is the length of the string (1 word)

|ticket types| is the length of the string (1 word)

Sample:

|  |  |
| --- | --- |
| Input | Output |
| REGISTER 7  Ani FAST\_TRACK  Bani REGULER  Cani FAST\_TRACK  Dani REGULER  Eni FAST\_TRACK  Fani REGULER  Gani REGULER  REMOVE 2  Bani  Dani  REGISTER 2  Ani FAST\_TRACK  Bani REGULER  CALL  CALL | No Name Ticket  1 Ani FAST\_TRACK  2 Cani FAST\_TRACK  3 Eni FAST\_TRACK  4 Bani REGULER  5 Dani REGULER  6 Fani REGULER  7 Gani REGULER  No Name Ticket  1 Ani FAST\_TRACK  2 Cani FAST\_TRACK  3 Eni FAST\_TRACK  4 Fani REGULER  5 Gani REGULER  No Name Ticket  1 Ani FAST\_TRACK  2 Cani FAST\_TRACK  3 Eni FAST\_TRACK  4 Ani FAST\_TRACK  5 Fani REGULER  6 Gani REGULER  7 Bani REGULER  Ani Cani Eni Ani got into the boat.  3 queues remaining.  Fani Gani Bani got into the boat.  0 queues remaining. |
|  |  |

A screenshot of a computer

Description automatically generated with medium confidence

\*Answer the case on Dev C++. The file extension to be uploaded is .cpp

-- Selamat Mengerjakan --