**บทที่ 1**

**บทนำ**

* 1. **ความเป็นมาของโครงงาน**

ในปัจจุบันเทคโนโลยีเข้ามามีบทบาทในทุก ๆ ด้าน พวกผมจึงคิดที่จะเขียนโปรแกรมห้องเช่าขึ้นมาเพื่อช่วยในการเก็บข้อมูลได้ง่ายขึ้นลดโอกาสการสูญหายของข้อมูล เพิ่มความเร็วในการจัดเก็บข้อมูล การค้นหาข้อมูล และการประมวลผลข้อมูลโดยการนำการจัดเก็บข้อมูลแบบ AVL Tree และการค้นหาข้อมูลแบบ Hashing มาใช้

**1.2 วัตถุประสงค์ของโครงงาน**

1. เพื่อพัฒนาโปรแกรมสำหรับผู้จัดการห้องเช่า

2. เพื่อสามารถจัดการการเก็บข้อมูล ค้นหาข้อมูล และประมวลผลข้อมูลได้รวดเร็วขึ้น

**1.3 ขอบเขตของโครงงาน**

เป็นโปรแกรมสำหรับจัดการห้องเช่าโดยการนำการเก็บข้อมูลแบบ AVL Tree และการค้นหาข้อมูลแบบ Hashing มาใช้โดยจะประกอบด้วย 6 เมนูดังนี้

1. **เมนูที่ 1 การใส่ข้อมูลผู้ที่พักอาศัย**

ใช้โครงสร้างการเก็บข้อมูลแบบ AVL จากฟังก์ชั่นที่สร้างขึ้นและรับข้อมูลผ่านทางแป้นพิมพ์

1. **เมนูที่ 2 การลบข้อมูลของผู้ที่พักอาศัย**

ลบข้อมูลที่เก็บในโครงสร้าง AVL ด้วยการค้นหาจาก เลขที่ห้องเพื่อเข้าไปลบข้อมูล

1. **เมนูที่ 3 อัพเดทข้อมูลหน่วยของค่าน้ำ-ค่าไฟของผู้พักอาศัย**

หาข้อมูลผู้พักอาศัยจากเลขที่ห้องและไปเพิ่มหน่วยของค่าน้ำค่าไฟเพื่อไว้ใช้คำนวณค่าเช่าต่อไป

1. **เมนูที่ 4 ปริ้นใบแจ้งชำระของผู้พักอาศัย**

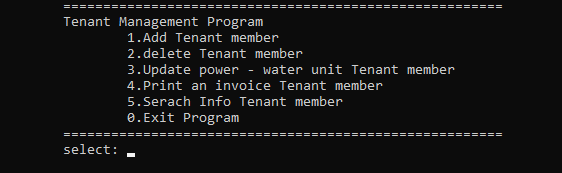
ฟังก์ชันนี้ใช้ปริ้นใบแจ้งชำระจากเลขที่ห้องที่ใส่ลงไป และจะบอกรายละเอียดค่าเช่าทุกอย่าง

1. **เมนูที่ 5 ค้นหาข้อมูลของผู้พักอาศัย**

ฟังก์ชันการค้นหาข้อมูล ใช้สำหรับผู้ใช้โปรแกรมที่ต้องการค้นหาข้อมูลผู้ที่เข้าพัก

1. **ออกจากโปรแกรม**

**ภาพการทำงานของโปรแกรม**



หน้านี้จะเป็นหน้าเมนูหลักเมื่อเริ่มเปิดโปรแกรมโดยจะมีเมนูให้เลือกอยู่ถึง 6 เมนูโดยให้พิมพ์ตัวเลขของเมนูที่ต้องการจะใช้งานโดยเลข 0 จะเป็นการออกจากโปรแกรม

โดยที่เมนูของโปรแกรมนี้มีทั้งหมด 6 เมนู ประกอบไปด้วย

**เมนูที่ 1 การใส่ข้อมูลผู้ที่พักอาศัย**

ใช้โครงสร้างการเก็บข้อมูลแบบ AVL จากฟังก์ชั่นที่สร้างขึ้นและรับข้อมูลผ่านทางแป้นพิมพ์

**เมนูที่ 2 การลบข้อมูลของผู้ที่พักอาศัย**

ลบข้อมูลที่เก็บในโครงสร้าง AVL ด้วยการค้นหาจาก เลขที่ห้องเพื่อเข้าไปลบข้อมูล

**เมนูที่ 3 อัพเดทข้อมูลหน่วยของค่าน้ำ-ค่าไฟของผู้พักอาศัย**

หาข้อมูลผู้พักอาศัยจากเลขที่ห้องและไปเพิ่มหน่วยของค่าน้ำค่าไฟเพื่อไว้ใช้คำนวณค่าเช่าต่อไป

**เมนูที่ 4 ปริ้นใบแจ้งชำระของผู้พักอาศัย**

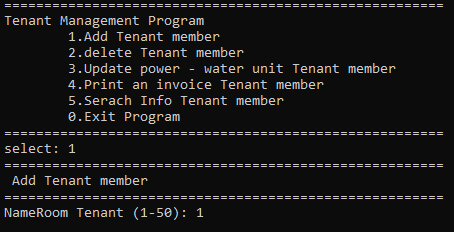
ฟังก์ชันนี้ใช้ปริ้นใบแจ้งชำระจากเลขที่ห้องที่ใส่ลงไป และจะบอกรายละเอียดค่าเช่าทุกอย่าง

**เมนูที่ 5 ค้นหาข้อมูลของผู้พักอาศัย**

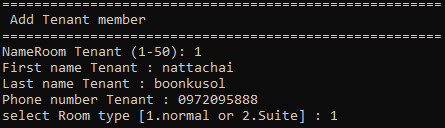
ฟังก์ชันการค้นหาข้อมูล ใช้สำหรับผู้ใช้โปรแกรมที่ต้องการค้นหาข้อมูลผู้ที่เข้าพัก

**ออกจากโปรแกรม**

**เมนูที่ 1 การใส่ข้อมูลผู้อยู่อาศัย (Add Tenant Member)**

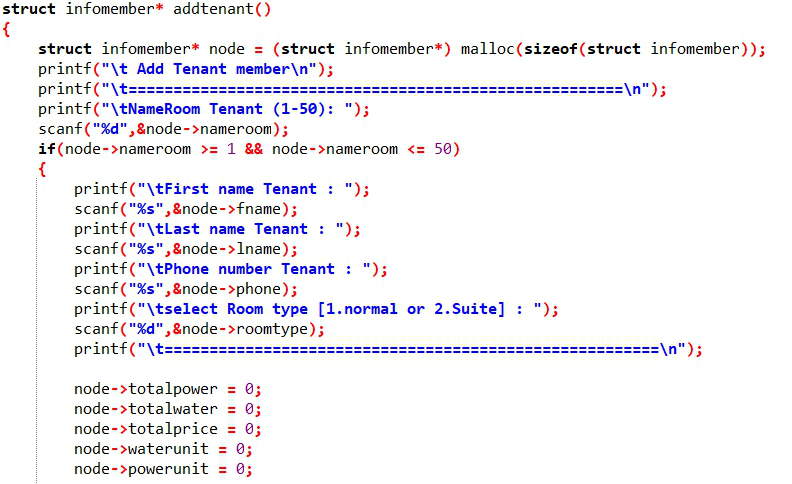


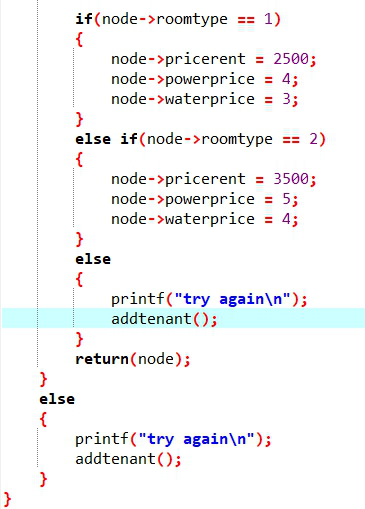
เมื่อพิมพ์เลข 1 ก็จะเข้ามาที่เมนู Add Tenant member โดยเมนูนี้จะทำหน้าที่เพิ่มรายละเอียดของผู้ที่จะเข้ามาอาศัยโปรแกรมจะเริ่มด้วยการกรอกเลขที่ห้อง หลังจากกรอกเลขที่ห้องก็จะให้กรอกเลขที่ห้องจะขึ้นให้ใส่ข้อมูลรายละเอียดต่างๆดังนี้



ให้กรอกชื่อจริง-นามสกุลของผู้พักอาศัย เบอร์ติดต่อและประเภทของห้องและทำการเก็บไว้ที่ฐานข้อมูลของ AVL การเก็บข้อมูล เลขที่ห้องจะเก็บเป็น INT ชื่อจริงนามสกุลและเบอร์โทรศัพท์เป็น STRING ประเภทห้องเป็น INT

**อธิบายฟังก์ชัน**

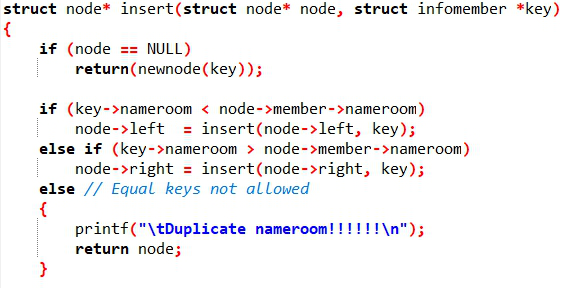


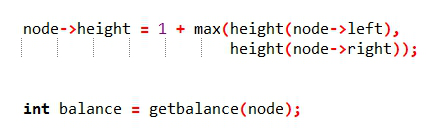


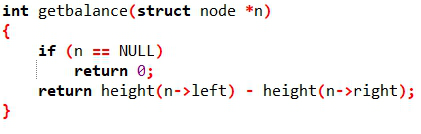
ฟังก์ชันนี้เป็นฟังก์ชันรับค่าข้อมูลทั้งหมดเพื่อไปใส่ใน AVL และเช็คประเภทห้องที่รับเข้ามาถ้าเป็นประเภทแรก จะราคาถูกกว่าประเภทที่ 2

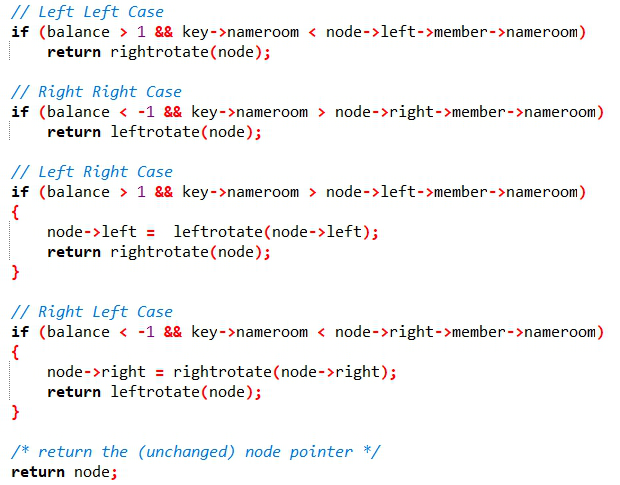
**ฟังก์ชันใส่ข้อมูลลง AVL**

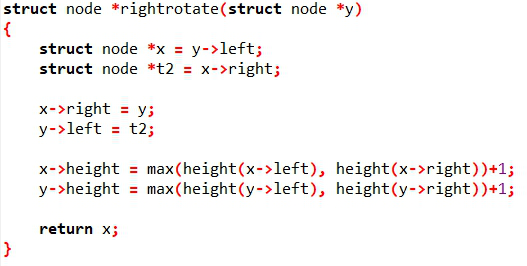
การทำงานคล้ายกับฟังก์ชันลบข้อมูลใน AVL คือการเช็คค่าจากเลขที่ห้องเช็คความสูงและทำให้สมดุลหลังจากนั้นถึงเพิ่มค่าเข้าไปในโหนดตามโค้ดด้านล่าง

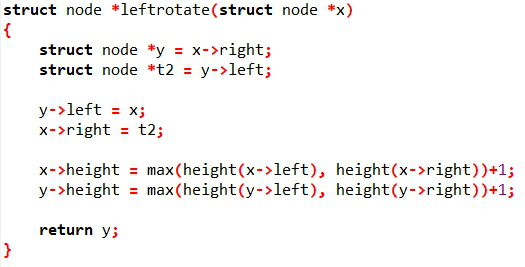




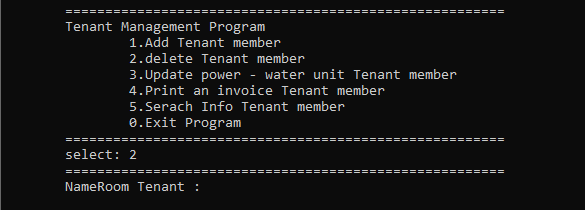




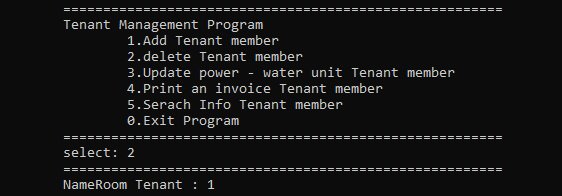




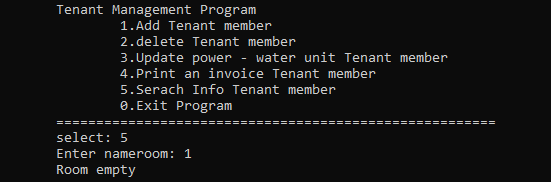
**เมนูที่ 2 ลบข้อมูลผู้ที่พักอาศัย (Delete Tenant member)**



เมื่อพิมพ์เลข 2 ในเมนูหลักจะปรากฏ เมนูที่ 2 ขึ้นคือเมนูลบข้อมูลด้วยการกรอกเลขที่ห้องพักที่ต้องการจะลบเหมือนดังรูปข้างต้น

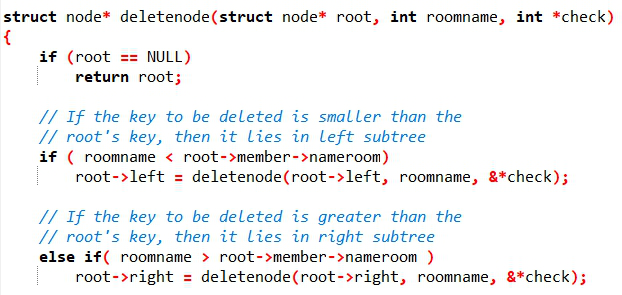


เมื่อกรอกเลขที่ห้องที่ต้องการจะลบแล้วกด enter โปรแกรมจะลบข้อมูลผู้พักอาศัยตามเลขที่ห้องที่ต้องการเลย

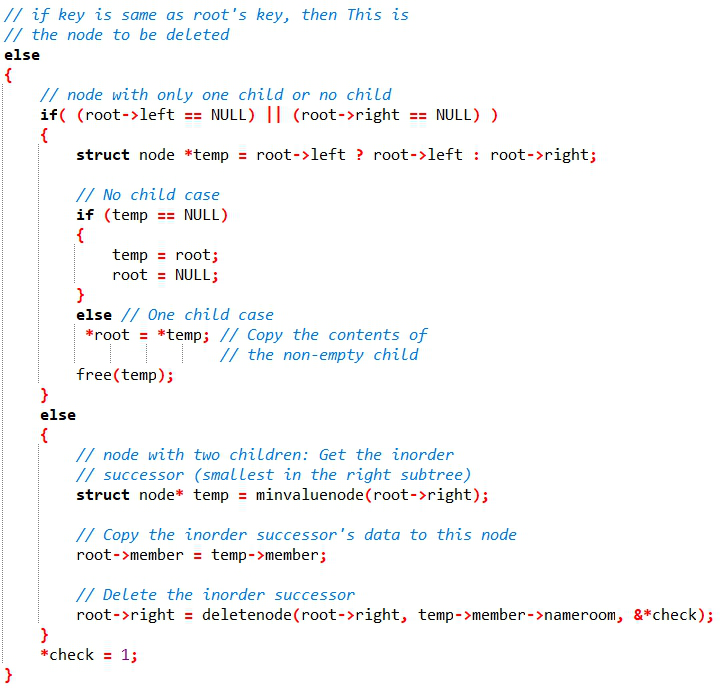


และเมื่อใช้ฟังก์ชั่นที่ 5 ในการค้นหาข้อมูลก็จะพบว่าข้อมูลของห้องที่ใส่ไว้ข้างต้นได้ถูกลบออกจากฐานข้อมูลเรียบร้อยแล้ว

**อธิบายฟังก์ชัน**



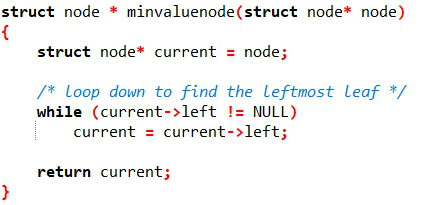
เป็นการค้นหาแบบ recursive ฟังก์ชัน เมื่อเจอค่ามากจะโยงไปขวา น้อยกว่าโยงไปซ้ายและทำฟังก์ชันซ้ำใหม่จนกว่าจะเจอค่า



เมื่อเจอค่าที่ต้องการจะลบจะเช็คว่าข้อมูลที่จะลบเป็นประเภทไหหนประเภทไม่มีลูก หรือ ลูก1 หรือมี ลูก 2 ก็จะใช้วิธีลบต่างกันไป

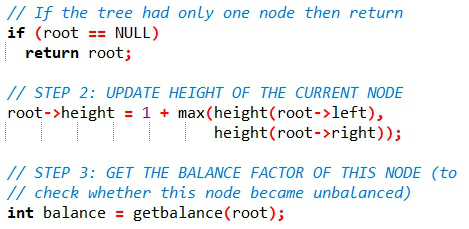
**ฟังก์ชันลบโหนดลูก 2**

จะลบโดยการหาค่าน้อยที่สุดจากโหนดฝั่งขวาตามรูปด้านบนโดยใช้ฟังก์ชันนี้ในการหา

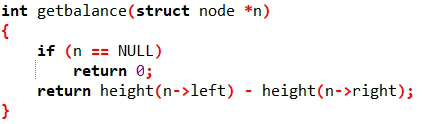


หลังจากนั้นจะทำการก็อปปี้ค่าขึ้นมาและทำการลบต่อไป

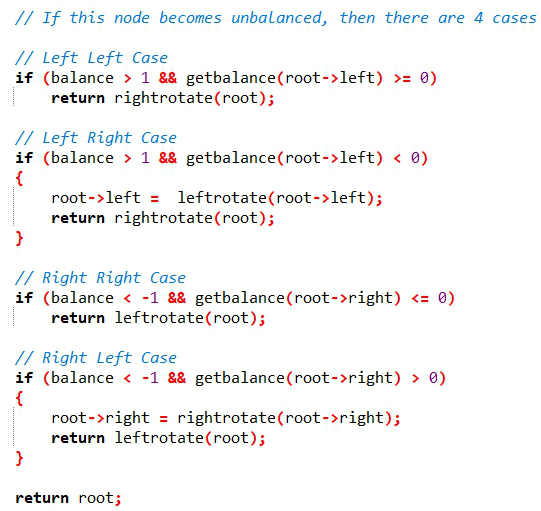
ฟังก์ชันหาความสมดุลและความสูงของโหนด



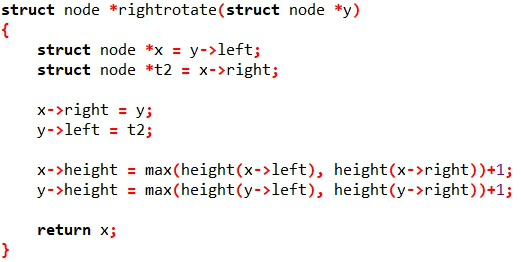
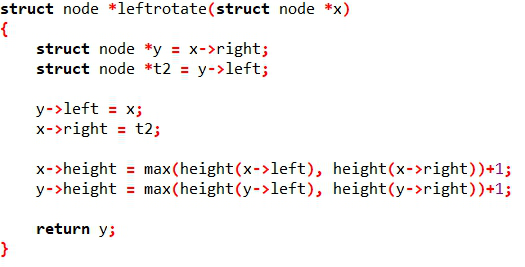
ฟังก์ชันนี้จะทำการเช็คความสูงของ AVL เพื่อที่เช็คว่าจะทำให้โครงสร้างต้นไม้สมดุลด้วยฟังก์ชัน getbalance



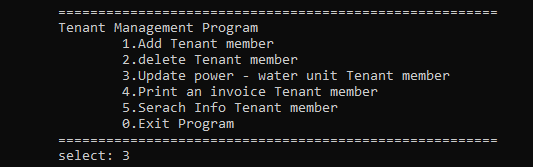
**ฟังก์ชันการหมุน**



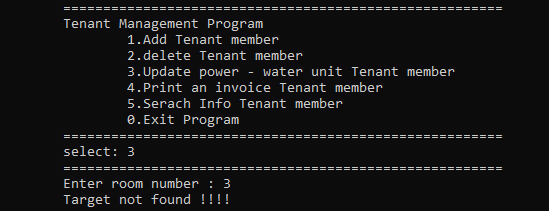
เมื่อรู้ค่าความสูงของทั้งซ้ายและขวาแล้วจะรู้ว่าสมดุลหรือไม่ ก็จะเช็คตามคำสั่งข้างต้นและจะทำการหมุนเพื่อให้ต้นไม้สมดุล



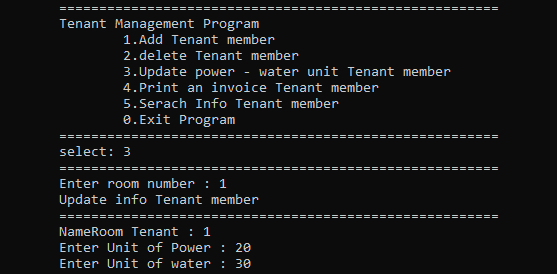
**เมนูที่ 3 อัพเดทข้อมูลหน่วยของค่าน้ำ-ค่าไฟของผู้พักอาศัย (Update power – water unit tenant member)**

****

เมื่อเลือกเมนูที่ 3 จะเข้ามาที่เมนูของการเพิ่มหน่วยของค่าน้ำและค่าไฟเพื่อใช้ในการคำนวณค่าเช่าของผู้ที่พักอาศัย

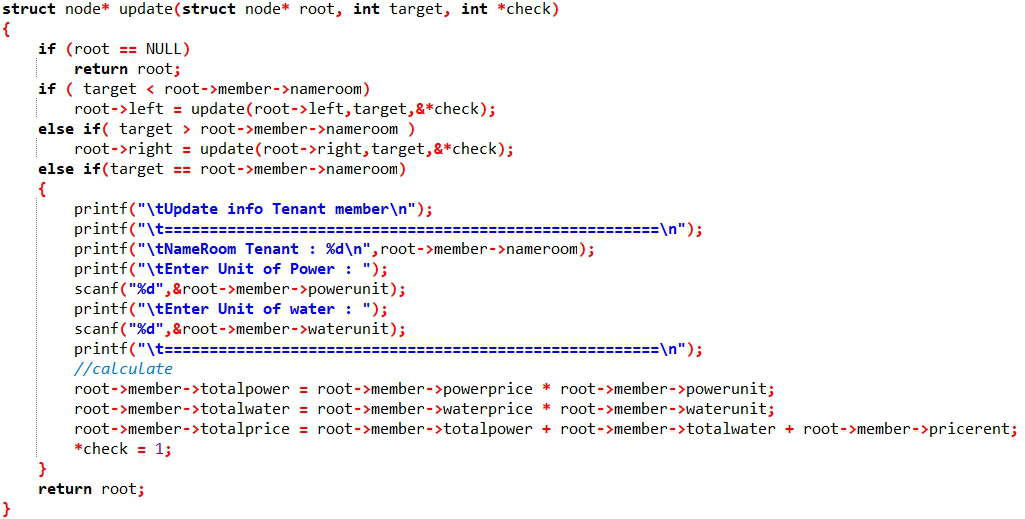


ต่อจากนั้นก็จะให้กรอกเลขที่ห้อง เมื่อกรอกเลขที่ห้องที่ไม่มีผู้ที่พักอาศัยหรือกรอกเลขที่ห้องผิดโปรแกรมก็จะแจ้งว่า ไม่เจอเป้าหมายที่ค้นหาและจะทำการเข้าสู่เมนูหลักใหม่อีกครั้ง



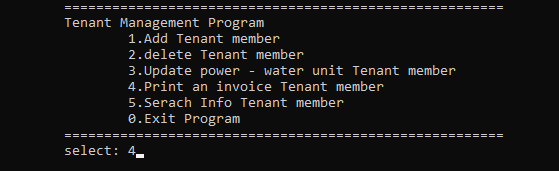
เมื่อกรอกเลขที่ห้องที่มีในฐานข้อมูลโปรแกรมก็จะให้ใส่หน่วยของค่าน้ำและค่าไฟและจบการทำงานของเมนูนี้

**อธิบายฟังก์ชัน**

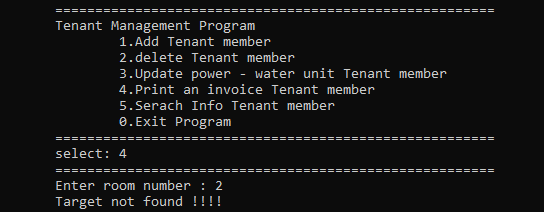


ทำการเช็คแบบ recusive เพื่อเช็คข้อมูลใน AVL ว่าตรงกับเลขห้องที่กรอกมาทางแป้นพิมพ์หรือไม่เมื่อท่องเข้าไปจนเจอข้อมูลตรงกับเลขที่ห้องก็จะทำการปริ้นรับค่าหน่วยของค่าน้ำ-และค่าไฟ หลังจากนั้นจะนำค่าทั้งหมดมาคำนวณโดยการนำหน่วยน้ำ-ไฟ คูณกับค่าน้ำ-ไฟต่อหน่วย และนำค่าน้ำ-ไฟ รวมกับค่าเช่าห้องจะได้เป็นค่าใช้จ่ายทั้งหมด

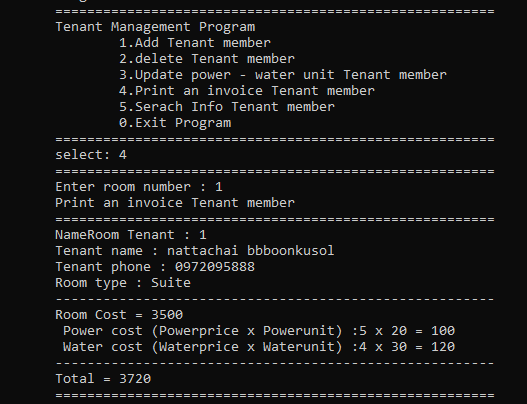
**เมนูที่ 4** **ปริ้นใบแจ้งชำระของผู้พักอาศัย (Print an invoice tenant member)**



เมื่อเลือกเมนูที่ 4 จากเมนูหลัก โปรแกรมจะเข้าสู่ฟังก์ชันการปริ้นข้อมูลทุกอย่างที่ต้องใช้ในการทำเป็นใบแจ้งชำระข้องผู้ที่พักอาศัย



เมื่อใส่เลขที่ห้องผิดหรือไม่มีในฐานข้อมูลจะขึ้นแจ้งเตือนหาข้อมูลไม่เจอเหมือนเมนูอื่นๆ



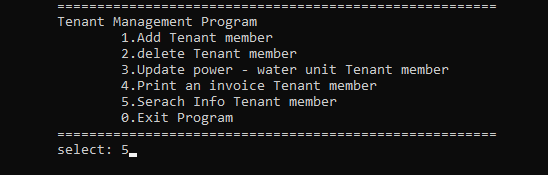
เมื่อใส่เลขห้องตรงตามที่มีในฐานข้อมูลโปรแกรมจะแจ้งรายละเอียดทั้งหมดและค่าเช่ารวมที่ต้องชำระ

**อธิบายฟังก์ชัน**

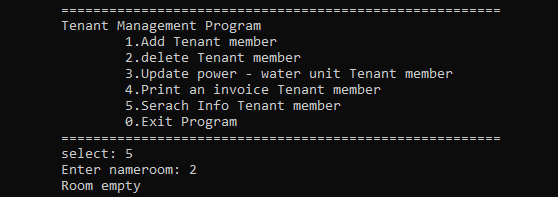


ฟังก์ชันในเมนูนี้จะรับค่าโครงสร้างของ AVL เป้าหมายเลขที่ห้องที่ต้องการค้นหา หลังจากนั้นฟังก์ชันจะเช็ค ว่า AVL ว่างหรือไม่ถ้าใช่จะทำการคืนค่าและกลับสู่เมนูหลัก หลังจากนั้นจะเช็คว่าเป้าหมายเลขที่ห้องมีค่าน้อยกว่าหรือมากกว่า ถ้าน้อยกว่าจะโยงไปทางซ้าย มากกว่าจะไปทางขวาและเมื่อหาเป้าหมายเจอจะทำการปริ้นตามรูปแบบที่ปริ้นตามข้างต้นโดยใช้ค่าจากใน AVL ที่ตรงกับหมายเลขห้องที่เป็นเป้าหมาย

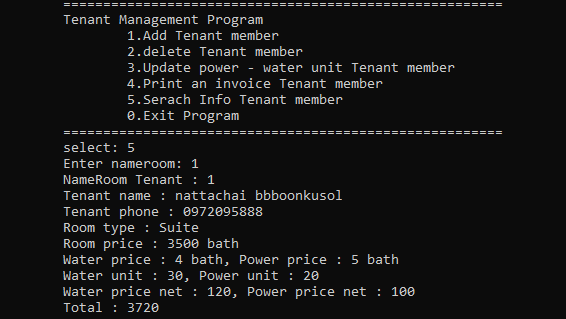
**เมนูที่ 5** **ค้นหาข้อมูลของผู้พักอาศัย (Search Info Tenant member)**

****

เมื่อเลือกเมนูที่ 5 จากเมนูหลักส่วนของเมนูนี้จะทำหน้าที่แจ้งข้อมูลตัวแปรทั้งหมดที่เก็บไว้ในฐานข้อมูลเพื่อให้ผู้ใช้งานเช็คค่าทั้งหมดก่อนจะทำการปริ้น

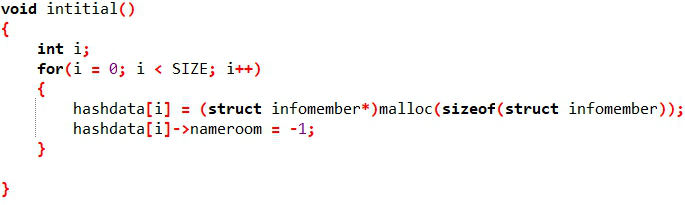


เมื่อใส่เลขที่ห้องที่ไม่มีข้อมูลของผู้พักอาศัยอยู่เลย โปรแกรมจะแจ้งว่าห้องนี้ว่างไม่มีผู้พักอาศัย

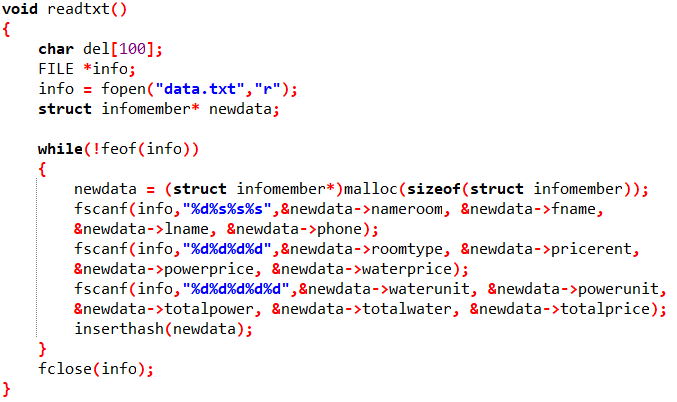


เมื่อกรอกเลขที่ห้องที่มีผู้พักอาศัยอยู่โปรแกรมจะแจ้งค่าตัวแปรทั้งหมดเพื่อเช็คความถูกต้องก่อนการปริ้น

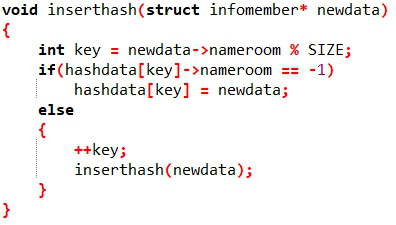
**อธิบายฟังก์ชัน**



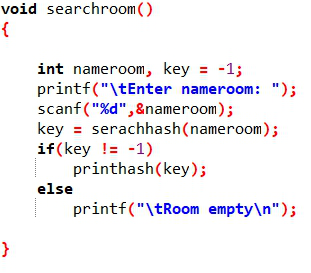
ฟังก์ชันนี้ใช้สำหรับกำหนดค่าของหมายเลขห้อง ให้มีค่าเป็น -1 ซึ่งถ้าหากหมายเลขห้องมีค่าเป็น -1 จะมีความหมายว่าข้อมูลตรงนี้ยังว่างอยู่สามารถนำข้อมูลมาใส่ได้



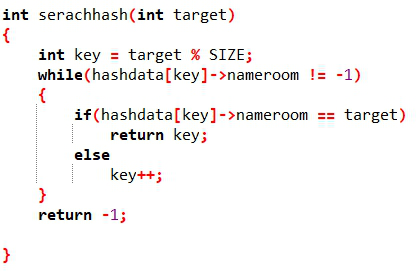
ฟังก์ชันนี้ใช้สำหรับอ่านข้อมูลจากฐานข้อมูลซึ่งจะเก็บข้อมูลไว้ในรูปแบบ .txt แล้ว

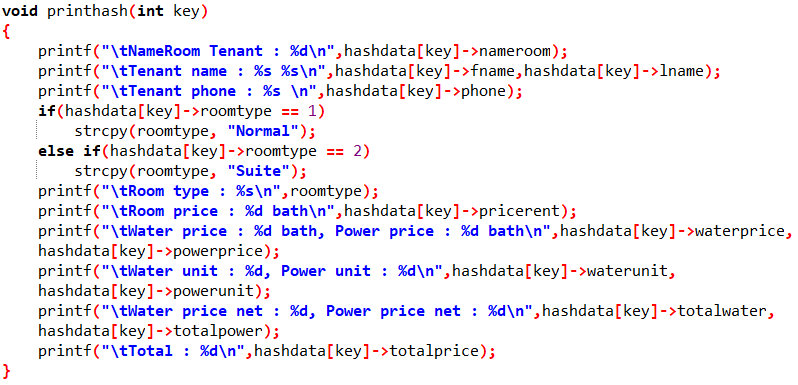


แล้วนำข้อมูลมาเก็บไว้ใน hashdata ซึ่งจะเก็บในรูปแบบ array of struct โดยการนำหมายเลขห้องมา % ขนาดของ table size ซึ่งกำหนดไว้เป็น 53 ซึ่งจะสามารถเก็บข้อมูลของห้องพักได้ 50 ห้อง เมื่อนำมา % เสร็จก็จะได้ address ของ hashdata แต่ถ้าเกิด address ซ้ำก็จะทำการ +1 ไปเรื่อยจนกว่าจะไม่ซ้ำกับข้อมูลที่มีอยู่แล้ว

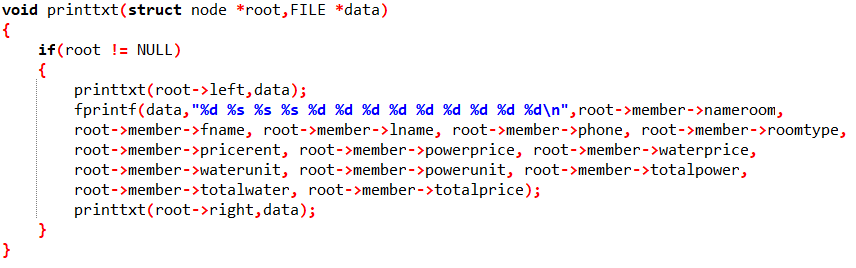


ฟังก์ชันนี้จะทำการรับค่าของหมายเลขห้องพักเพื่อนำข้อมูลไปค้นหาว่ามีข้อมูลคนมาเข้าพักไหมโดยจะเรียกฟังก์ชัน serachhash ถ้าเกิดมีข้อมูลอยู่จะทำการส่ง address ของข้อมูลของห้องที่ค้นหากลับมาและจะทำการเรียกฟังก์ชัน printhash โดยการส่ง address เข้าไปเพื่อทำการพิมพ์ข้อมูลของหมายเลขห้องพักที่ค้นหาออกมา แต่ถ้าไม่มีจะส่ง -1 กลับมา



ฟังก์ชันนี้จะทำการหาตำแหน่งของหมายเลขห้องพักที่ส่งมาถ้าหากมีข้อมูลก็จะส่ง address ของข้อมูลกลับไป แต่ถ้าไม่มีจะส่ง -1 กลับไปแทน

ฟังก์ชันนี้จะรับค่า address ของหมายเลขห้องพักจากฟังก์ชัน serachroom และจะนำพิมพ์ข้อมูลต่างๆของหมายเลขห้องพักนี้ออกมาทั้งหมดทางหน้าจอ

ฟังก์ชันนี้จะทำการพิมพ์ข้อมูลจากฐานข้อมูลที่เป็น AVL ลงมาเป็น .txt เพื่อในมาใช้ในการค้นหาข้อมูลแบบ hashing

**ภาคผนวก**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<string.h>**

**#define SIZE 53**

**int max(int a, int b);**

**char roomtype[10];**

**struct infomember \*hashdata[SIZE];**

**struct node**

**{**

**struct infomember \*member;**

**struct node \*left, \*right;**

**int height;**

**};**

**struct infomember**

**{**

**char fname[30], lname[30], phone[10];**

**int nameroom, roomtype, pricerent, powerprice, waterprice, waterunit, powerunit, totalpower, totalwater, totalprice;**

**};**

**int height(struct node \*n)**

**{**

**if (n == NULL)**

**return 0;**

**return n->height;**

**}**

**int max(int a, int b)**

**{**

**return (a > b)? a : b;**

**}**

**struct node\* newnode(struct infomember \*key)**

**{**

**struct node\* node = (struct node\*) malloc(sizeof(struct node));**

**node->member = key;**

**node->left = NULL;**

**node->right = NULL;**

**node->height = 1; // new node is initially added at leaf**

**return(node);**

**}**

**struct node \*rightrotate(struct node \*y)**

**{**

**struct node \*x = y->left;**

**struct node \*t2 = x->right;**

**x->right = y;**

**y->left = t2;**

**x->height = max(height(x->left), height(x->right))+1;**

**y->height = max(height(y->left), height(y->right))+1;**

**return x;**

**}**

**struct node \*leftrotate(struct node \*x)**

**{**

**struct node \*y = x->right;**

**struct node \*t2 = y->left;**

**y->left = x;**

**x->right = t2;**

**x->height = max(height(x->left), height(x->right))+1;**

**y->height = max(height(y->left), height(y->right))+1;**

**return y;**

**}**

**int getbalance(struct node \*n)**

**{**

**if (n == NULL)**

**return 0;**

**return height(n->left) - height(n->right);**

**}**

**struct node\* insert(struct node\* node, struct infomember \*key)**

**{**

**if (node == NULL)**

**return(newnode(key));**

**if (key->nameroom < node->member->nameroom)**

**node->left = insert(node->left, key);**

**else if (key->nameroom > node->member->nameroom)**

**node->right = insert(node->right, key);**

**else // Equal keys not allowed**

**{**

**printf("\tDuplicate nameroom!!!!!!\n");**

**return node;**

**}**

**node->height = 1 + max(height(node->left),**

**height(node->right));**

**int balance = getbalance(node);**

**// Left Left Case**

**if (balance > 1 && key->nameroom < node->left->member->nameroom)**

**return rightrotate(node);**

**// Right Right Case**

**if (balance < -1 && key->nameroom > node->right->member->nameroom)**

**return leftrotate(node);**

**// Left Right Case**

**if (balance > 1 && key->nameroom > node->left->member->nameroom)**

**{**

**node->left = leftrotate(node->left);**

**return rightrotate(node);**

**}**

**// Right Left Case**

**if (balance < -1 && key->nameroom < node->right->member->nameroom)**

**{**

**node->right = rightrotate(node->right);**

**return leftrotate(node);**

**}**

**/\* return the (unchanged) node pointer \*/**

**return node;**

**}**

**struct node \* minvaluenode(struct node\* node)**

**{**

**struct node\* current = node;**

**/\* loop down to find the leftmost leaf \*/**

**while (current->left != NULL)**

**current = current->left;**

**return current;**

**}**

**struct node\* deletenode(struct node\* root, int roomname, int \*check)**

**{**

**if (root == NULL)**

**return root;**

**// If the key to be deleted is smaller than the**

**// root's key, then it lies in left subtree**

**if ( roomname < root->member->nameroom)**

**root->left = deletenode(root->left, roomname, &\*check);**

**// If the key to be deleted is greater than the**

**// root's key, then it lies in right subtree**

**else if( roomname > root->member->nameroom )**

**root->right = deletenode(root->right, roomname, &\*check);**

**// if key is same as root's key, then This is**

**// the node to be deleted**

**else**

**{**

**// node with only one child or no child**

**if( (root->left == NULL) || (root->right == NULL) )**

**{**

**struct node \*temp = root->left ? root->left : root->right;**

**// No child case**

**if (temp == NULL)**

**{**

**temp = root;**

**root = NULL;**

**}**

**else // One child case**

**\*root = \*temp; // Copy the contents of**

**// the non-empty child**

**free(temp);**

**}**

**else**

**{**

**// node with two children: Get the inorder**

**// successor (smallest in the right subtree)**

**struct node\* temp = minvaluenode(root->right);**

**// Copy the inorder successor's data to this node**

**root->member = temp->member;**

**// Delete the inorder successor**

**root->right = deletenode(root->right, temp->member->nameroom, &\*check);**

**}**

**\*check = 1;**

**}**

**// If the tree had only one node then return**

**if (root == NULL)**

**return root;**

**// STEP 2: UPDATE HEIGHT OF THE CURRENT NODE**

**root->height = 1 + max(height(root->left),**

**height(root->right));**

**// STEP 3: GET THE BALANCE FACTOR OF THIS NODE (to**

**// check whether this node became unbalanced)**

**int balance = getbalance(root);**

**// If this node becomes unbalanced, then there are 4 cases**

**// Left Left Case**

**if (balance > 1 && getbalance(root->left) >= 0)**

**return rightrotate(root);**

**// Left Right Case**

**if (balance > 1 && getbalance(root->left) < 0)**

**{**

**root->left = leftrotate(root->left);**

**return rightrotate(root);**

**}**

**// Right Right Case**

**if (balance < -1 && getbalance(root->right) <= 0)**

**return leftrotate(root);**

**// Right Left Case**

**if (balance < -1 && getbalance(root->right) > 0)**

**{**

**root->right = rightrotate(root->right);**

**return leftrotate(root);**

**}**

**return root;**

**}**

**struct node\* update(struct node\* root, int target, int \*check)**

**{**

**if (root == NULL)**

**return root;**

**if ( target < root->member->nameroom)**

**root->left = update(root->left,target,&\*check);**

**else if( target > root->member->nameroom )**

**root->right = update(root->right,target,&\*check);**

**else if(target == root->member->nameroom)**

**{**

**printf("\tUpdate info Tenant member\n");**

**printf("\t=======================================================\n");**

**printf("\tNameRoom Tenant : %d\n",root->member->nameroom);**

**printf("\tEnter Unit of Power : ");**

**scanf("%d",&root->member->powerunit);**

**printf("\tEnter Unit of water : ");**

**scanf("%d",&root->member->waterunit);**

**printf("\t=======================================================\n");**

**//calculate**

**root->member->totalpower = root->member->powerprice \* root->member->powerunit;**

**root->member->totalwater = root->member->waterprice \* root->member->waterunit;**

**root->member->totalprice = root->member->totalpower + root->member->totalwater + root->member->pricerent;**

**\*check = 1;**

**}**

**return root;**

**}**

**struct node\* printinv(struct node\* root, int target, int \*check)**

**{**

**if (root == NULL)**

**return root;**

**if ( target < root->member->nameroom)**

**root->left = printinv(root->left,target,&\*check);**

**else if( target > root->member->nameroom )**

**root->right = printinv(root->right,target,&\*check);**

**else if(target == root->member->nameroom)**

**{**

**printf("\tPrint an invoice Tenant member\n");**

**printf("\t=======================================================\n");**

**printf("\tNameRoom Tenant : %d\n",root->member->nameroom);**

**printf("\tTenant name : %s %s\n",root->member->fname,root->member->lname);**

**printf("\tTenant phone : %s \n",root->member->phone);**

**if(root->member->roomtype == 1)**

**strcpy(roomtype, "Normal");**

**else**

**strcpy(roomtype, "Suite");**

**printf("\tRoom type : %s\n",roomtype);**

**printf("\t-------------------------------------------------------\n");**

**printf("\tRoom Cost = %d\n",root->member->pricerent);**

**printf("\t Power cost (Powerprice x Powerunit) :%d x %d = %d \n",root->member->powerprice,root->member->powerunit,**

**root->member->totalpower);**

**printf("\t Water cost (Waterprice x Waterunit) :%d x %d = %d \n",root->member->waterprice,root->member->waterunit,**

**root->member->totalwater);**

**printf("\t-------------------------------------------------------\n");**

**printf("\tTotal = %d\n",root->member->totalprice);**

**printf("\t=======================================================\n");**

**\*check = 1;**

**}**

**return root;**

**}**

**void inorder(struct node \*root)**

**{**

**if(root != NULL)**

**{**

**inorder(root->left);**

**printf("%d\n", root->member->nameroom);**

**inorder(root->right);**

**}**

**}**

**int printmenu()**

**{**

**int select;**

**printf("\t=======================================================\n");**

**printf("\tTenant Management Program\n"**

**"\t\t1.Add Tenant member\n"**

**"\t\t2.delete Tenant member\n"**

**"\t\t3.Update power - water unit Tenant member\n"**

**"\t\t4.Print an invoice Tenant member\n"**

**"\t\t5.Serach Info Tenant member\n"**

**"\t\t0.Exit Program\n"**

**"\t=======================================================\n"**

**"\tselect: ");**

**scanf("%d",&select);**

**return select;**

**}**

**/\*struct infomember\* readtxt()**

**{**

**FILE \*info;**

**info = fopen("data.txt","r");**

**while(!feof(info))**

**{**

**struct infomember\* node = (struct infomember\*) malloc(sizeof(struct infomember));**

**fscanf(info,"%d%s%s%s%d",&node->nameroom, &node->fname, &node->lname, &node->phone, &node->roomtype);**

**root = insert(node, node->nameroom);**

**}**

**}\*/**

**struct infomember\* addtenant()**

**{**

**struct infomember\* node = (struct infomember\*) malloc(sizeof(struct infomember));**

**printf("\t Add Tenant member\n");**

**printf("\t=======================================================\n");**

**printf("\tNameRoom Tenant (1-50): ");**

**scanf("%d",&node->nameroom);**

**if(node->nameroom >= 1 && node->nameroom <= 50)**

**{**

**printf("\tFirst name Tenant : ");**

**scanf("%s",&node->fname);**

**printf("\tLast name Tenant : ");**

**scanf("%s",&node->lname);**

**printf("\tPhone number Tenant : ");**

**scanf("%s",&node->phone);**

**printf("\tselect Room type [1.normal or 2.Suite] : ");**

**scanf("%d",&node->roomtype);**

**printf("\t=======================================================\n");**

**node->totalpower = 0;**

**node->totalwater = 0;**

**node->totalprice = 0;**

**node->waterunit = 0;**

**node->powerunit = 0;**

**if(node->roomtype == 1)**

**{**

**node->pricerent = 2500;**

**node->powerprice = 4;**

**node->waterprice = 3;**

**}**

**else if(node->roomtype == 2)**

**{**

**node->pricerent = 3500;**

**node->powerprice = 5;**

**node->waterprice = 4;**

**}**

**else**

**{**

**printf("try again\n");**

**addtenant();**

**}**

**return(node);**

**}**

**else**

**{**

**printf("try again\n");**

**addtenant();**

**}**

**}**

**//----------------------------------------------------- Hashing---------------------------------------------------**

**void intitial()**

**{**

**int i;**

**for(i = 0; i < SIZE; i++)**

**{**

**hashdata[i] = (struct infomember\*)malloc(sizeof(struct infomember));**

**hashdata[i]->nameroom = -1;**

**strcpy(hashdata[i]->fname, "a");**

**strcpy(hashdata[i]->lname, "a");**

**strcpy(hashdata[i]->phone, "a");**

**hashdata[i]->roomtype = 1;**

**}**

**}**

**void inserthash(struct infomember\* newdata)**

**{**

**int key = newdata->nameroom % SIZE;**

**if(hashdata[key]->nameroom == -1)**

**hashdata[key] = newdata;**

**else**

**{**

**++key;**

**inserthash(newdata);**

**}**

**}**

**void readtxt()**

**{**

**char del[100];**

**FILE \*info;**

**info = fopen("data.txt","r");**

**struct infomember\* newdata;**

**while(!feof(info))**

**{**

**newdata = (struct infomember\*)malloc(sizeof(struct infomember));**

**fscanf(info,"%d%s%s%s%d%d%d%d%d%d%d%d%d",&newdata->nameroom, &newdata->fname, &newdata->lname, &newdata->phone, &newdata->roomtype, &newdata->pricerent, &newdata->powerprice, &newdata->waterprice, &newdata->waterunit, &newdata->powerunit, &newdata->totalpower, &newdata->totalwater, &newdata->totalprice);**

**inserthash(newdata);**

**}**

**fclose(info);**

**}**

**void printtxt(struct node \*root,FILE \*data)**

**{**

**if(root != NULL)**

**{**

**printtxt(root->left,data);**

**fprintf(data,"%d %s %s %s %d %d %d %d %d %d %d %d %d\n",root->member->nameroom, root->member->fname, root->member->lname, root->member->phone, root->member->roomtype, root->member->pricerent, root->member->powerprice, root->member->waterprice, root->member->waterunit, root->member->powerunit, root->member->totalpower, root->member->totalwater, root->member->totalprice);**

**printtxt(root->right,data);**

**}**

**}**

**int serachhash(int target)**

**{**

**int key = target % SIZE;**

**while(hashdata[key]->nameroom != -1)**

**{**

**if(hashdata[key]->nameroom == target)**

**return key;**

**else**

**key++;**

**}**

**return -1;**

**}**

**void printhash(int key)**

**{**

**printf("\tNameRoom Tenant : %d\n",hashdata[key]->nameroom);**

**printf("\tTenant name : %s %s\n",hashdata[key]->fname,hashdata[key]->lname);**

**printf("\tTenant phone : %s \n",hashdata[key]->phone);**

**if(hashdata[key]->roomtype == 1)**

**strcpy(roomtype, "Normal");**

**else if(hashdata[key]->roomtype == 2)**

**strcpy(roomtype, "Suite");**

**printf("\tRoom type : %s\n",roomtype);**

**printf("\tRoom price : %d bath\n",hashdata[key]->pricerent);**

**printf("\tWater price : %d bath, Power price : %d bath\n",hashdata[key]->waterprice, hashdata[key]->powerprice);**

**printf("\tWater unit : %d, Power unit : %d\n",hashdata[key]->waterunit, hashdata[key]->powerunit);**

**printf("\tWater price net : %d, Power price net : %d\n",hashdata[key]->totalwater, hashdata[key]->totalpower);**

**printf("\tTotal : %d\n",hashdata[key]->totalprice);**

**}**

**void searchroom()**

**{**

**int nameroom, key = -1;**

**printf("\tEnter nameroom: ");**

**scanf("%d",&nameroom);**

**key = serachhash(nameroom);**

**if(key != -1)**

**printhash(key);**

**else**

**printf("\tRoom empty\n");**

**}**

**/\*void printhash()**

**{**

**int i;**

**for(i=0; i<SIZE; i++)**

**{**

**if(hashdata[i]->nameroom != -1)**

**{**

**printf("\tNameRoom Tenant : %d\n",hashdata[i]->nameroom);**

**printf("\tTenant name : %s %s\n",hashdata[i]->fname,hashdata[i]->lname);**

**printf("\tTenant phone : %s \n",hashdata[i]->phone);**

**if(hashdata[i]->roomtype == 1)**

**strcpy(roomtype, "Normal");**

**else**

**strcpy(roomtype, "Suite");**

**printf("\tRoom type : %s\n",roomtype);**

**}**

**}**

**}\*/**

**//----------------------------------------------------- Main ----------------------------------------------------------**

**int main()**

**{**

**FILE \*data;**

**int check = 0;**

**int select=-1;**

**struct node \*root = NULL;**

**while(select != 0)**

**{**

**select = printmenu();**

**if(select == 1)**

**{**

**printf("\t=======================================================\n");**

**data = fopen("data.txt","w");**

**root = insert(root, addtenant());**

**printtxt(root,data);**

**fclose(data);**

**}**

**else if(select == 2)**

**{**

**printf("\t=======================================================\n");**

**data = fopen("data.txt","w");**

**int nameroom;**

**printf("\tNameRoom Tenant : ");**

**scanf("%d",&nameroom);**

**root = deletenode(root, nameroom, &check);**

**if(check == 0)**

**printf("\tTarget not found !!!! \n");**

**check = 0;**

**printtxt(root,data);**

**fclose(data);**

**}**

**else if(select == 3) //update**

**{**

**printf("\t=======================================================\n");**

**data = fopen("data.txt","w");**

**int target;**

**printf("\tEnter room number : ");**

**scanf("%d",&target);**

**root = update(root,target,&check);**

**if(check == 0)**

**printf("\tTarget not found !!!! \n");**

**check = 0;**

**printtxt(root,data);**

**fclose(data);**

**}**

**else if(select == 4) //printinvoice**

**{**

**printf("\t=======================================================\n");**

**int target;**

**printf("\tEnter room number : ");**

**scanf("%d",&target);**

**root = printinv(root,target,&check);**

**if(check == 0)**

**printf("\tTarget not found !!!! \n");**

**check = 0;**

**}**

**else if(select == 5) //serach**

**{**

**intitial();**

**readtxt();**

**searchroom();**

**}**

**}**

**printf("\t=======================================================\n");**

**printf("\tGood bye, See you agin......\n");**

**return 0;**

**}**