```
#include <stdio.h>
#include <stdlib.h>

typedef struct item
{
    int key;
    int value;
    struct item *point;
} hash;

struct hashtable_item
{
    int flag;
    hash *data;
};

int size = 0;
int max = 10;
struct hashtable_item *array;

void insert(int key, int value)
{
    int index = key % max;
    int i = index, h = 1;

    hash *new_item = (hash *)malloc(sizeof(hash));
    new_item->key = key;
    new_item->value = value;

    while (array[i].flag == 1)
    {
        i = (key + (h * h)) % max;
        h++;
        if (i == index)
        {
            printf("\n Hash table is full, cannot insert any more item
\n");
            return;
        }
    }

    array[i].flag = 1;
    array[i].data = new_item;
    size++;
    printf("\n Key (%d) has been inserted \n", key);
}

void remove_element(int key)
{
    int index = key % max;
    int i = index, h = 1;

    while (array[i].flag != 0)
    {

        if (array[i].flag == 1 && array[i].data->key == key)
```

```
        {

            array[i].flag = 0;
            array[i].data = NULL;
            size--;
            printf("\n Key (%d) has been removed \n", key);
            return;
        }

        i = (key + (h * h)) % max;
        h++;
        if (i == index)
            break;
    }
    printf("\n This key does not exist \n");
}

void display()
{
    for (int i = 0; i < max; i++)
    {
        printf("%d\t", i);
        if (array[i].data != NULL)
            printf("%d %d", array[i].data->key, array[i].data->value);
        printf("\n");
    }
}

void search(int key)
{
    int index = key % max;
    int i = index, h = 1;

    while (array[i].flag != 0)
    {

        if (array[i].flag == 1 && array[i].data->key == key)
        {
            int m = array[i].data->value;
            printf("\n Key (%d) has been found whose value is %d\n",
key, m);
            return;
        }

        i = (i + (h * h)) % max;
        h++;
        if (i == index)
            break;
    }
    printf("\n This key does not exist \n");
}

typedef struct Node
{
    int data;
    struct Node *left, *right, *mid;
} Node;
```

```
Node *newNode(int data)
{
    Node *temp = (Node *)malloc(sizeof(Node));
    temp->data = data;
    temp->left = temp->right = temp->mid = NULL;
    return temp;
}

void findMinMax(Node *node, int *min, int *max, int *middle, int hd)
{
    if (node == NULL)
        return;

    if (hd < *min)
    {
        *min = hd;
    }
    else if (hd > *min && hd < *max)
    {
        *middle = hd;
    }
    else if (hd > *max)
    {
        *max = hd;
    }

    findMinMax(node->left, min, max, middle, hd - 1);
    findMinMax(node->mid, min, max, middle, hd);
    findMinMax(node->right, min, max, middle, hd + 1);
}

void printVerticalLine(Node *node, int line_no, int hd)
{
    if (node == NULL)
        return;

    if (hd == line_no)
        printf("%d ", node->data);

    printVerticalLine(node->left, line_no, hd - 1);
    printVerticalLine(node->mid, line_no, hd);
    printVerticalLine(node->right, line_no, hd + 1);
}

void verticalOrder(Node *root)
{
    int min = 0, max = 0, middle = 0;
    findMinMax(root, &min, &max, &middle, 0);

    for (int line_no = min; line_no <= max; line_no++)
    {
        printVerticalLine(root, line_no, 0);
        printf("\n");
    }
}
void main()
```

```
{
    int choice, key, value, n;
    char c;
    array = (struct hashtable_item *)malloc(max * sizeof(struct
hashtable_item));
    for (int i = 0; i < max; i++)
    {
        array[i].flag = 0;
        array[i].data = NULL;
    }
    printf("how may keys you want to enter\n");
    int x;
    scanf("%d", &x);
    printf("Inserting element in Hashtable\n");
    while (x--)
    {
        printf("Enter key and value-:\t");
        scanf("%d %d", &key, &value);
        insert(key, value);
    }
    while (1)
    {
        printf("\n-------------------------\n");
        printf("MENU-: \n1.Display Hashtable"
               "\n2.Removing item from the Hashtable"
               "\n3.search Hashtable"
               "\n4.for exit"
               "\n\n Please enter your choice-:");

        scanf("%d", &choice);

        switch (choice)
        {

        case 1:
            display();
            break;

        case 2:

            printf("Deleting in Hashtable \n Enter the key to delete-
:");
            scanf("%d", &key);
            remove_element(key);
            display();
            break;

        case 3:
            printf("Enter key\n");
            scanf("%d", &key);
            search(key);
            break;
        case 4:
            exit(0);
            break;

        default:
```

```
            printf("Wrong Input\n");
        }
    }

    /* Node *root = newNode(1);
    root->left = newNode(2);
    root->mid = newNode(3);
    root->right = newNode(4);
    root->left->left = newNode(5);
    root->left->mid = newNode(6);
    root->left->right = newNode(7);
    root->mid->left = newNode(8);
    root->mid->mid = newNode(9);
    root->mid->right = newNode(10);
    root->right->left = newNode(11);
    root->right->mid = newNode(12);
    root->right->right = newNode(13);
    root->right->right->left = newNode(14);
    root->right->right->mid = newNode(15);
    root->right->right->right = newNode(16);

    printf("Vertical order traversal is \n");
    verticalOrder(root);*/
}
```