Question number 1(A) :-

#include<iostream>

using namespace std;

class HashTable

{

int\* table;

int Size;

int count;

int getHashValue(int num) // Private member function of

{

return (num % Size);

}

public:

HashTable(int x)

{

count = 0;

Size = x;

table = new int[x];

for (int i = 0; i < x; i++)

{

table[i] = -1;

}

}

~HashTable()

{

delete table;

table = NULL;

}

bool isEmpty()

{

if (count == 0)

return 1;

return 0;

}

bool isFull()

{

if (Size == count)

return 1;

return 0;

}

double loadFactor()

{

return(double(count) / double(Size));

}

void insertion(int val)

{

int key;

int i = 0;

key = getHashValue(val);

if (isFull()) {

cout << "Array is full....." << endl;

return;

}

while (table[key] != -1)

{

key++;

key = key % Size;

}

table[key] = val;

count++;

cout << endl << "Insertion Succesfully." << endl;

}

void Display()

{

for (int i = 0; i < Size; i++)

{

if(table[i]!=-1)

cout << i + 1 << " --> " << table[i] << endl;

else {

cout << i + 1 << " --> " << endl;

}

}

}

bool Search(int val)

{

int key = getHashValue(val);

for (int i = 0; i < 4; i++)

{

if (table[key] == val)

return 1;

key++;

if (key >= Size)

key = key % Size;

}

return 0;

}

bool Delete(int val)

{

int key = getHashValue(val);

if (isEmpty())

return 0;

if (Search(val))

{

while (table[key] != val)

{

key++;

if (key >= Size)

key = key % Size;

}

table[key] = -1;

count--;

return 1;

}

else

return 0;

}

};

int main()

{

int num;

int x;

int flag;

cout << "Enter size of Table : ";

cin >> num;

HashTable obj(num);

while (true)

{

system("cls");

cout << "1) Insertion " << endl;

cout << "2) Display " << endl;

cout << "3) Search " << endl;

cout << "4) Delete " << endl;

cout << "0) Exit" << endl;

cin >> x;

cout << endl;

switch (x)

{

case 1:

cout << "Enter Value : ";

cin >> num;

obj.insertion(num);

system("pause");

break;

case 2:

obj.Display();

system("pause");

break;

case 3:

cout << "Enter Value To Search : ";

cin >> num;

flag = obj.Search(num);

if (flag)

cout << "Entered Value Is Found In Hash Table." << endl;

else

cout << "Entered Value Is Not Found In Hash Table." << endl;

system("pause");

break;

case 4:

cout << "Enter Value To Delete : ";

cin >> num;

flag = obj.Delete(num);

if (flag)

cout << "Entered Value Is Deleted In Hash Table." << endl;

else

cout << "Entered Value Is Not Deleted In Hash Table." << endl;

system("pause");

break;

case 0:

return 0;

default:

cout << "Wrong Input!" << endl;

system("pause");

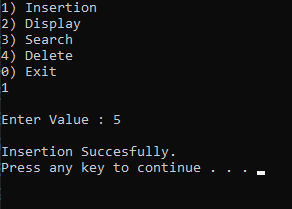
}

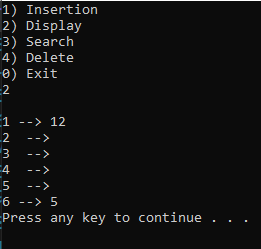
}

return 0;

}

Result :-





Question Number 1(B) :-

#include<iostream>

using namespace std;

class HashTable

{

int step;

int\* table;

int Size;

int count;

int getHashValue(int num) // Private member function of

{

return (num % Size);

}

bool prime(int val)

{

int count = 0;

for (int i = 2; i <= val; i++)

{

if (val % i == 0)

count++;

}

if (count > 1)

return 0;

else

return 1;

}

public:

HashTable(int x)

{

count = 0;

Size = x;

table = new int[x];

for (int i = 0; i < x; i++)

{

table[i] = -1;

}

for (int i = 2; i < x; i++) {

if (prime(i)) {

if (x % i == 0) {

step = i;

break;

}

}

}

}

~HashTable()

{

delete table;

table = NULL;

}

bool isEmpty()

{

if (count == 0)

return 1;

return 0;

}

bool isFull()

{

if (Size == count)

return 1;

return 0;

}

void Incre()

{

int\* dum = table;

int size = Size;

Size = Size \* 2;

table = new int[size];

for (int i = 0; i < Size; i++)

{

table[i] = -1;

}

count = 0;

for (int i = 0; i < size; i++)

{

if (dum[i] != -1)

{

insertion(dum[i]);

}

}

delete[] dum;

dum = NULL;

}

double loadFactor()

{

return(double(count) / double(Size));

}

void insertion(int val)

{

int key;

int i = 0;

key = getHashValue(val);

if (isFull()) {

cout << "Array is full....." << endl;

return;

}

while (table[key] != -1)

{

key += step;

key = key % Size;

}

table[key] = val;

count++;

cout << endl << "Insertion Succesfully." << endl;

if (loadFactor() > 0.7)

{

Incre();

}

}

void Display()

{

for (int i = 0; i < Size; i++)

{

if(table[i]!=-1)

cout << i + 1 << " --> " << table[i] << endl;

else {

cout << i + 1 << " --> " << endl;

}

}

}

bool Search(int val)

{

int key = getHashValue(val);

for (int i = 0; i < 4; i++)

{

if (table[key] == val)

return 1;

key++;

if (key >= Size)

key = key % Size;

}

return 0;

}

bool Delete(int val)

{

int key = getHashValue(val);

if (isEmpty())

return 0;

if (Search(val))

{

while (table[key] != val)

{

key++;

if (key >= Size)

key = key % Size;

}

table[key] = -1;

count--;

return 1;

}

else

return 0;

}

};

int main()

{

int num;

int x;

int flag;

cout << "Enter size of Table : ";

cin >> num;

HashTable obj(num);

while (true)

{

system("cls");

cout << "1) Insertion " << endl;

cout << "2) Display " << endl;

cout << "3) Search " << endl;

cout << "4) Delete " << endl;

cout << "0) Exit" << endl;

cin >> x;

cout << endl;

switch (x)

{

case 1:

cout << "Enter Value : ";

cin >> num;

obj.insertion(num);

system("pause");

break;

case 2:

obj.Display();

system("pause");

break;

case 3:

cout << "Enter Value To Search : ";

cin >> num;

flag = obj.Search(num);

if (flag)

cout << "Entered Value Is Found In Hash Table." << endl;

else

cout << "Entered Value Is Not Found In Hash Table." << endl;

system("pause");

break;

case 4:

cout << "Enter Value To Delete : ";

cin >> num;

flag = obj.Delete(num);

if (flag)

cout << "Entered Value Is Deleted In Hash Table." << endl;

else

cout << "Entered Value Is Not Deleted In Hash Table." << endl;

system("pause");

break;

case 0:

return 0;

default:

cout << "Wrong Input!" << endl;

system("pause");

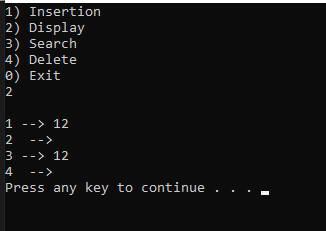
}

}

return 0;

}

Result :-



Question Number 1(C) :-

#include<iostream>

using namespace std;

class HashTable

{

int step;

int\* table;

int Size;

int count;

int getHashValue(int num) // Private member function of

{

return (num % Size);

}

bool prime(int val)

{

int count = 0;

for (int i = 2; i <= val; i++)

{

if (val % i == 0)

count++;

}

if (count > 1)

return 0;

else

return 1;

}

bool isEmpty()

{

if (count == 0)

return 1;

return 0;

}

bool isFull()

{

if (Size == count)

return 1;

return 0;

}

void Incre()

{

int\* dum = table;

int size = Size;

Size = Size \* 2;

table = new int[size];

for (int i = 0; i < Size; i++)

{

table[i] = -1;

}

count = 0;

for (int i = 0; i < size; i++)

{

if (dum[i] != -1)

{

insertion(dum[i]);

}

}

delete[] dum;

dum = NULL;

}

public:

HashTable(int x)

{

count = 0;

Size = x;

table = new int[x];

for (int i = 0; i < x; i++)

{

table[i] = -1;

}

for (int i = 2; i < x; i++) {

if (prime(i)) {

if (x % i == 0) {

step = i;

break;

}

}

}

}

~HashTable()

{

delete table;

table = NULL;

}

double loadFactor()

{

return(double(count) / double(Size));

}

void insertion(int val)

{

int key;

int i = 0;

key = getHashValue(val);

if (isFull()) {

cout << "Array is full....." << endl;

return;

}

while (table[key] != -1)

{

key += (i \* i);

if (key >= Size)

key = key % Size;

i++;

}

table[key] = val;

count++;

cout << endl << "Insertion Succesfully." << endl;

if (loadFactor() > 0.7)

{

Incre();

}

}

void Display()

{

for (int i = 0; i < Size; i++)

{

if(table[i]!=-1)

cout << i + 1 << " --> " << table[i] << endl;

else {

cout << i + 1 << " --> " << endl;

}

}

}

bool Search(int val)

{

int key = getHashValue(val);

for (int i = 0; i < 4; i++)

{

if (table[key] == val)

return 1;

key++;

if (key >= Size)

key = key % Size;

}

return 0;

}

bool Delete(int val)

{

int key = getHashValue(val);

if (isEmpty())

return 0;

if (Search(val))

{

while (table[key] != val)

{

key++;

if (key >= Size)

key = key % Size;

}

table[key] = -1;

count--;

return 1;

}

else

return 0;

}

};

int main()

{

int num;

int x;

int flag;

cout << "Enter size of Table : ";

cin >> num;

HashTable obj(num);

while (true)

{

system("cls");

cout << "1) Insertion " << endl;

cout << "2) Display " << endl;

cout << "3) Search " << endl;

cout << "4) Delete " << endl;

cout << "0) Exit" << endl;

cin >> x;

cout << endl;

switch (x)

{

case 1:

cout << "Enter Value : ";

cin >> num;

obj.insertion(num);

system("pause");

break;

case 2:

obj.Display();

system("pause");

break;

case 3:

cout << "Enter Value To Search : ";

cin >> num;

flag = obj.Search(num);

if (flag)

cout << "Entered Value Is Found In Hash Table." << endl;

else

cout << "Entered Value Is Not Found In Hash Table." << endl;

system("pause");

break;

case 4:

cout << "Enter Value To Delete : ";

cin >> num;

flag = obj.Delete(num);

if (flag)

cout << "Entered Value Is Deleted In Hash Table." << endl;

else

cout << "Entered Value Is Not Deleted In Hash Table." << endl;

system("pause");

break;

case 0:

return 0;

default:

cout << "Wrong Input!" << endl;

system("pause");

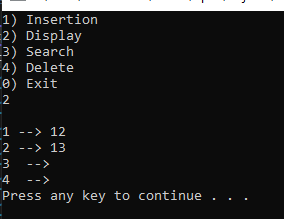
}

}

return 0;

}

Result :-



Question Number 2(A) :-

#include<iostream>

#include<time.h>

using namespace std;

struct node

{

int data;

node\* next;

node()

{

data = -1;

next = NULL;

}

node(int name) :node()

{

this->data = name;

}

};

class HashTable

{

int Size;

int count;

int getHashValue(int num) // Private member function of

{

return (num % Size);

}

public:

node\*\* table;

HashTable(int x)

{

count = 0;

Size = x;

table = new node \* [x];

for (int i = 0; i < Size; i++)

{

table[i] = NULL;

}

}

~HashTable()

{

}

double loadFactor()

{

return(double(count) / double(Size));

}

void insertion(int val)

{

node\* newnode = new node(val);

int key;

key = getHashValue(val);

node\* temp = table[key];

if (temp == NULL)

{

temp = newnode;

table[key] = temp;

return;

}

while (temp->next != NULL)

{

temp = temp->next;

}

temp->next = newnode;

count++;

return;

}

void Display()

{

node\* temp;

for (int i = 0; i < Size; i++)

{

cout << " -> ";

temp = table[i];

while (temp != NULL)

{

cout << temp->data;

temp = temp->next;

if (temp != NULL)

cout << " -> ";

}

cout << endl;

}

}

bool Search(int val)

{

node\* temp;

int i = 0;

int key = getHashValue(val);

temp = table[key];

while (temp && temp->data!=val)

{

temp = temp->next;

}

if (temp) {

return 1;

}

return 0;

}

bool Delete(int val)

{

node\* temp;

int key = getHashValue(val);

temp = table[key];

if (Search(val))

{

while (temp->next != NULL)

{

temp = temp->next;

}

temp->data = -1;

count--;

return 1;

}

else

return 0;

}

};

int main()

{

int num;

int x;

int flag;

cout << "Enter size of Table : ";

cin >> num;

HashTable obj(num);

while (true)

{

system("cls");

cout << "1) Insertion " << endl;

cout << "2) Display " << endl;

cout << "3) Search " << endl;

cout << "4) Delete " << endl;

cout << "0) Exit" << endl;

cin >> x;

cout << endl;

switch (x)

{

case 1:

cout << "Enter Value : ";

cin >> num;

obj.insertion(num);

system("pause");

break;

case 2:

obj.Display();

system("pause");

break;

case 3:

cout << "Enter Value To Search : ";

cin >> num;

flag = obj.Search(num);

if (flag)

cout << "Entered Value Is Found In Hash Table." << endl;

else

cout << "Entered Value Is Not Found In Hash Table." << endl;

system("pause");

break;

case 4:

cout << "Enter Value To Delete : ";

cin >> num;

flag = obj.Delete(num);

if (flag)

cout << "Entered Value Is Deleted In Hash Table." << endl;

else

cout << "Entered Value Is Not Deleted In Hash Table." << endl;

system("pause");

break;

case 0:

return 0;

default:

cout << "Wrong Input!" << endl;

system("pause");

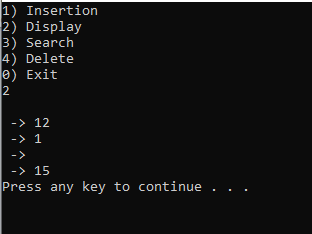
}

}

return 0;

}

Result :-



Question number 2(B) :-

#include<iostream>

#include<time.h>

using namespace std;

class HashTable

{

int SubBu;

int buck;

int Size;

int count;

int SubHash(int num) {

return ((num % SubBu));

}

int getHashValue(int num) // Private member function of

{

return ((num % buck));

}

bool isEmpty()

{

if (count == 0)

return 1;

return 0;

}

bool isFull()

{

if (Size == count)

return 1;

return 0;

}

public:

int\*\* table;

HashTable(int x)

{

Rai:

cout << "Enter number of buckets : ";

cin >> buck;

if (x % buck != 0) {

cout << "Number of buckets Must be divisible to Total Number...." << endl;

system("pause");

system("cls");

goto Rai;

}

count = 0;

Size = x;

int cal;

cal = x / buck;

SubBu = cal;

table = new int \* [buck];

for (int i = 0; i < buck; i++)

{

table[i] = new int [cal];

}

for (int i = 0; i < buck; i++) {

for (int k = 0; k < SubBu; k++) {

table[i][k] = -1;

}

}

}

~HashTable()

{

}

double loadFactor()

{

return(double(count) / double(Size));

}

void insertion(int val)

{

if (isFull()) {

return;

}

int key, key2;

key = key2 = val;

key = getHashValue(key);

for (int i = 1; i <= buck; i++) {

key2 = SubHash(key2);

for (int k = key2; k < SubBu; k++) {

if (table[key][key2] == -1) {

table[key][key2] = val;

count++;

return;

}

key2 = key2 + 3;;

key2 = key2 % SubBu;

}

key = key + 3;

key = key % buck;

}

cout << "Couldn't Find the empty Slot....." << endl;

}

void Display()

{

for (int i = 0; i < buck; i++) {

for (int k = 0; k < SubBu; k++) {

if (table[i][k] != -1)

cout << "-> " << table[i][k];

}

cout << endl;

}

}

bool Search(int val)

{

int x = getHashValue(val);

int x1;

for (int i = 1; i <= buck; i++) {

x1 = SubHash(val);

for (int k = x1; k <= SubBu + x1; k++) {

if (table[x][x1] == val)

return 1;

x1 = x1 + 3;

x1 = x1 % SubBu;

}

x = x + 3;

x = x % buck;

}

return 0;

}

bool Delete(int val)

{

int x, x1;

if (Search(val))

{

x = getHashValue(val);

for (int i = 1; i <= buck; i++) {

x1 = SubHash(val);

for (int k = x1; k <= SubBu + x1; k++) {

if (table[x][x1] == val) {

table[x][x1] = -1;

return 1;

}

x1++;

x1 = x1 % SubBu;

}

x++;

x = x % buck;

}

}

return 0;

}

};

int main()

{

int arr[16] = { 17,26,15,19,11,43,75,19,35,45,55,9,10,21,61,23 };

int num;

int x;

int flag;

cout << "Enter size of Table : ";

cin >> num;

HashTable obj(num);

while (true)

{

system("cls");

cout << "1) Insertion " << endl;

cout << "2) Display " << endl;

cout << "3) Search " << endl;

cout << "4) Delete " << endl;

cout << "0) Exit" << endl;

cin >> x;

cout << endl;

switch (x)

{

case 1:

/\*cout << "Enter Value : ";

cin >> num;\*/

for (int i = 0; i < 16; i++) {

obj.insertion(arr[i]);

}

system("pause");

break;

case 2:

obj.Display();

system("pause");

break;

case 3:

cout << "Enter Value To Search : ";

cin >> num;

flag = obj.Search(num);

if (flag)

cout << "Entered Value Is Found In Hash Table." << endl;

else

cout << "Entered Value Is Not Found In Hash Table." << endl;

system("pause");

break;

case 4:

cout << "Enter Value To Delete : ";

cin >> num;

flag = obj.Delete(num);

if (flag)

cout << "Entered Value Is Deleted In Hash Table." << endl;

else

cout << "Entered Value Is Not Deleted In Hash Table." << endl;

system("pause");

break;

case 0:

return 0;

default:

cout << "Wrong Input!" << endl;

system("pause");

}

}

return 0;

}

Result :-

