InterPlanetary File System (IPFS)

Created by:

Afnan Hassan (i22-0991)

Mehboob Ali (i22-1208)

Rana Bilal Akbar (i22-1094)

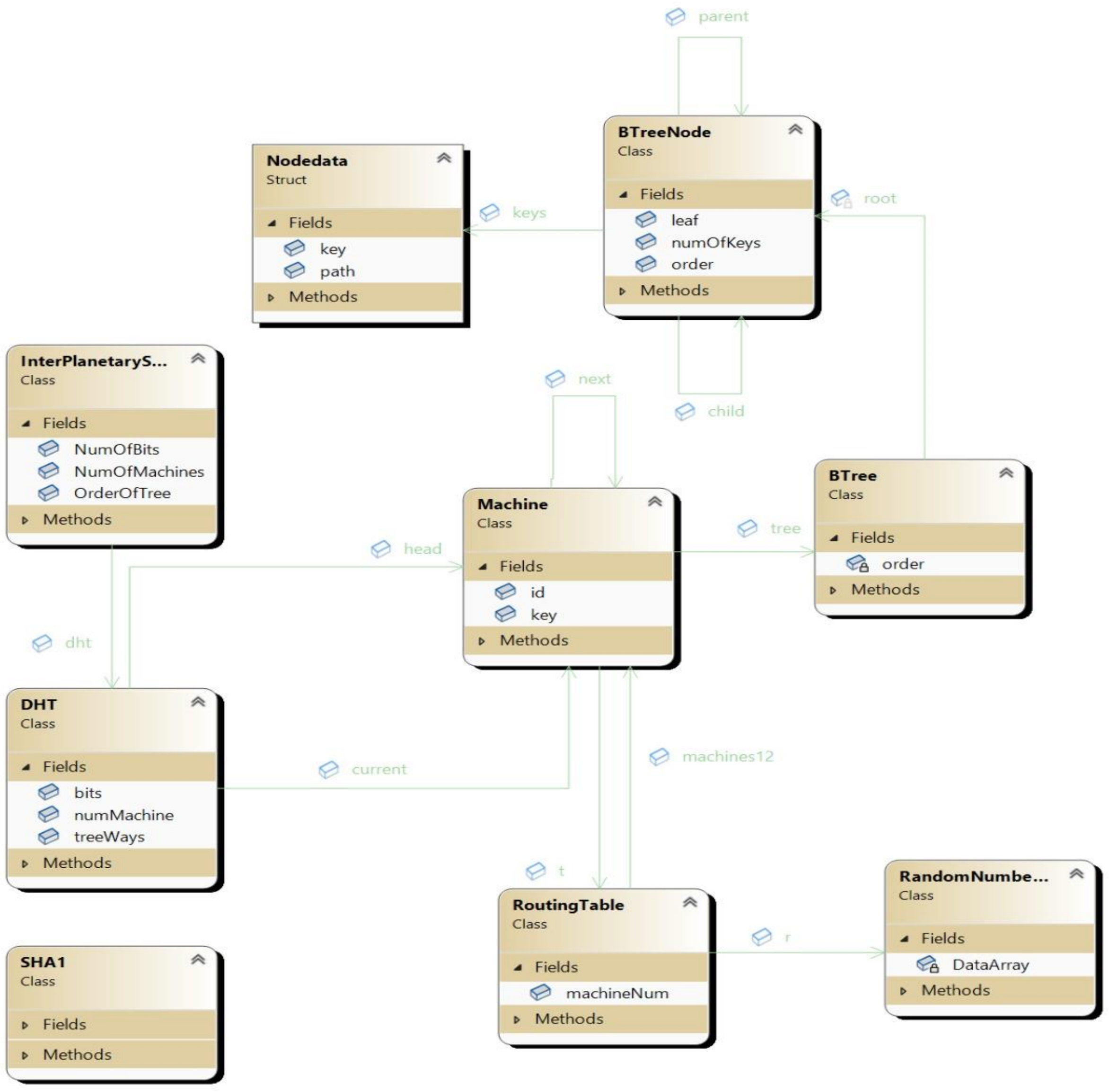
Table of Contents

Introduction: 3
Class Diagram: 3
Usage: 4
Assigning IDs to Machines: 6
Accessing the Machines: 8
Insertion of the File: 9
Deletion of File: 10
Display Routing Table: 11
Change current machine: 11
Add Machine: 12
Remove Machine: 13
Display B tree: 14
Work Division: 15

Introduction:

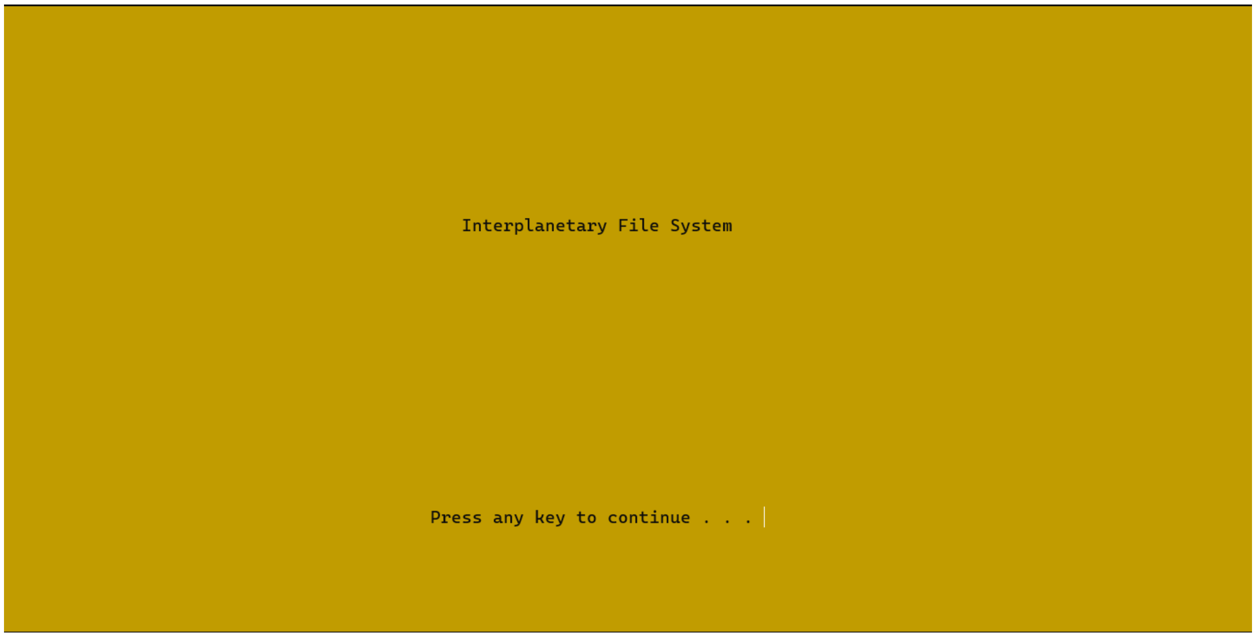
We were assigned to create a filing system like the IPFS using the concepts we were taught during the class. We had to use a Distributed Hash Tables (DHT) created using the circular linked list and created machines at each node. We then had to store the files inserted in the machines in the form of the B tree. To generate hash, we used SHA1 hash function that read the data from the file and then converted it into a key that was converted to a hex number after extracting exact bits (entered by the user) from the hex string SHA1 generated. Then the program finds the required machine after using the routing table and then inserts the file in B tree of appropriate machine. Moreover, options to display the B tree of required machine and delete the file stored in the B tree by specifying the key to that file are included in the code. Users can also choose to enter an additional machine to the circular linked list or delete any existing machine without disrupting the functionality of the DHT.

Class Diagram:



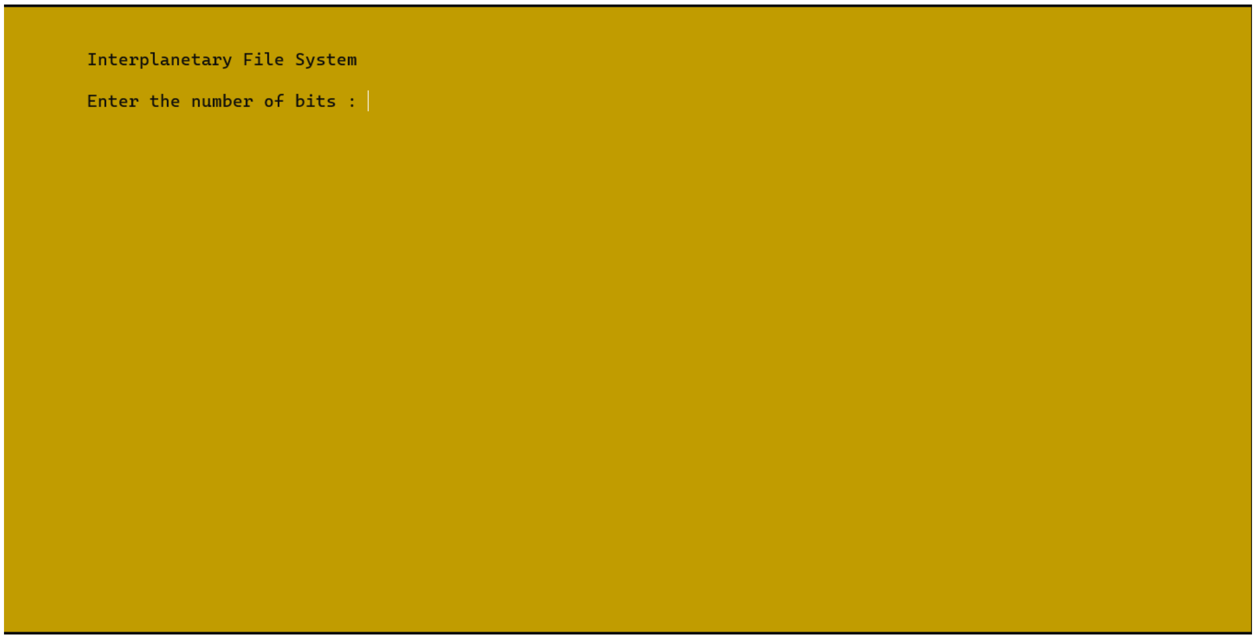
Usage:

Welcome Screen



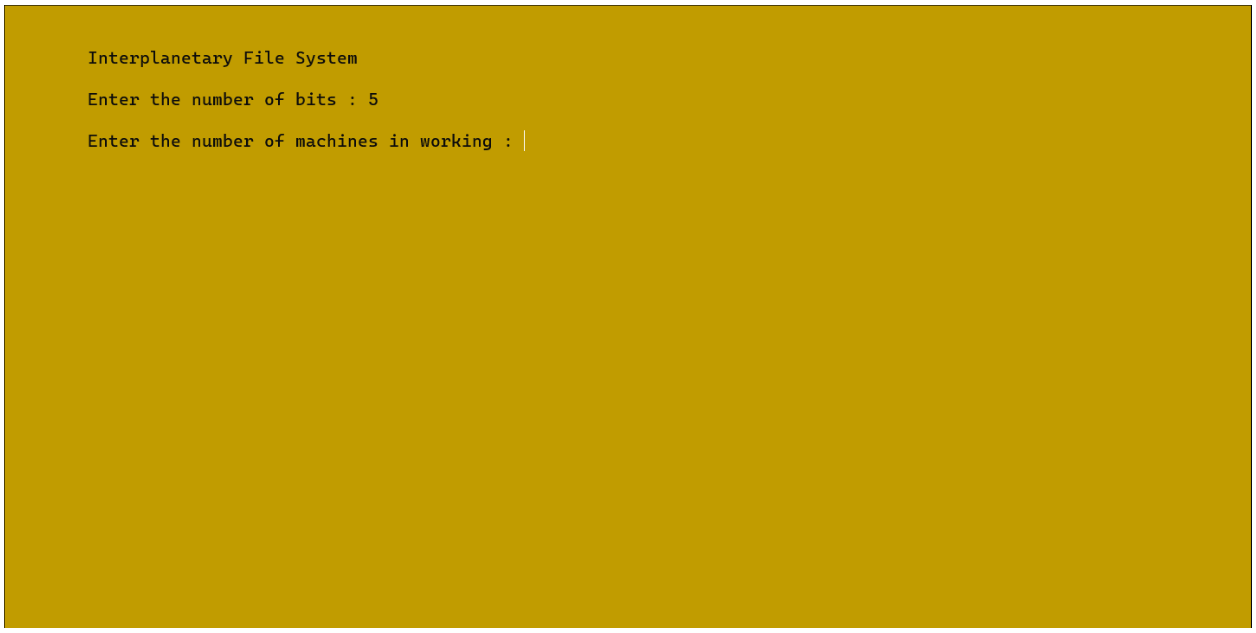
```
Interplanetary File System

Press any key to continue . . .
```

- Here the user is asked to input the number of the bits according to which the identifier space will be created



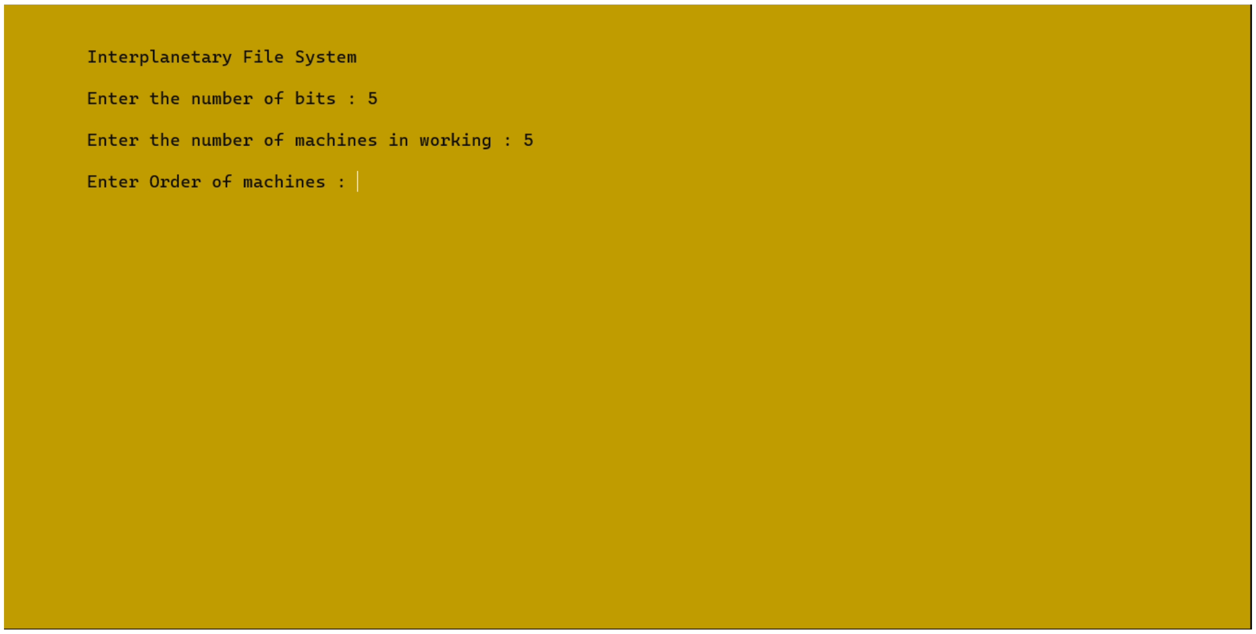
```
Interplanetary File System

Enter the number of bits :
```

- Now the user if asked to input the number of machines they wish to add to DHT



```
Interplanetary File System

Enter the number of bits : 5

Enter the number of machines in working :
```

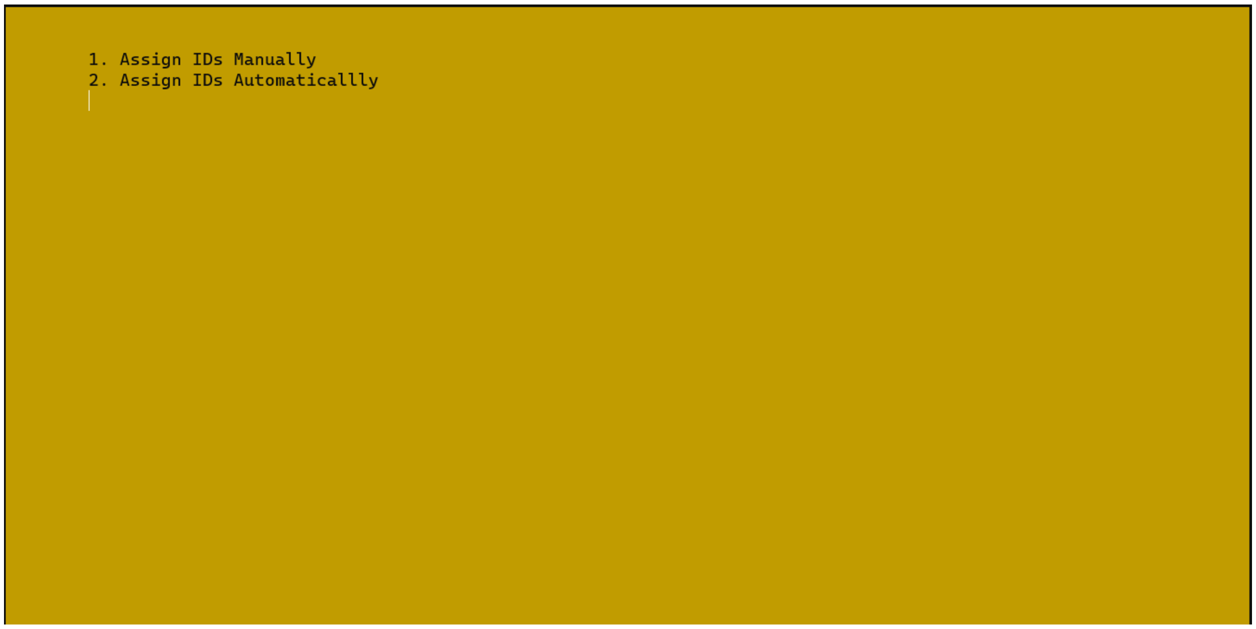
- Then the user if asked to input the max order of the B trees



```
Interplanetary File System

Enter the number of bits : 5

Enter the number of machines in working : 5

Enter Order of machines :
```

According to the user’s input a Circular linked list is created with specified Identifier space and Machines.

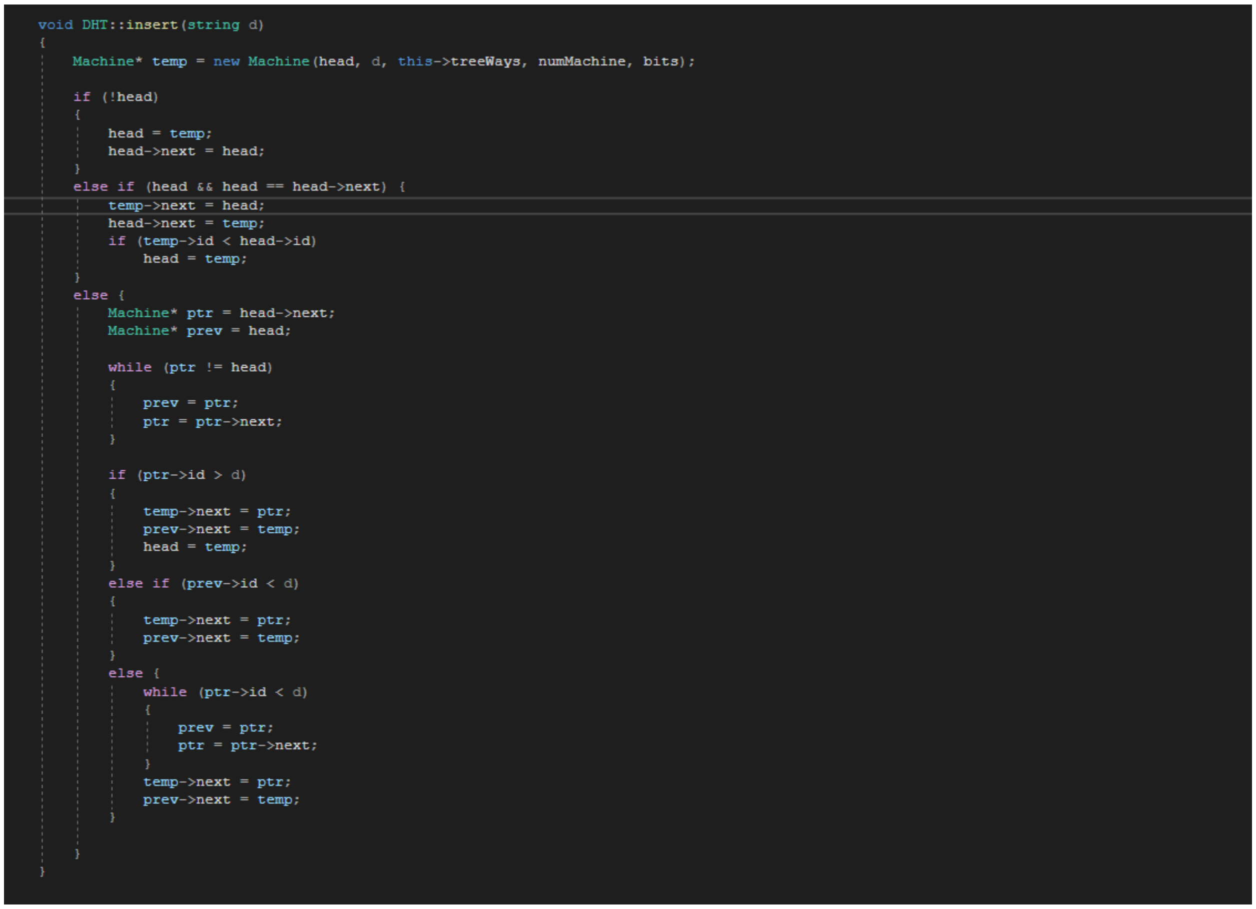
Assigning IDs to Machines:



```
1. Assign IDs Manually
2. Assign IDs Automaticallly
```

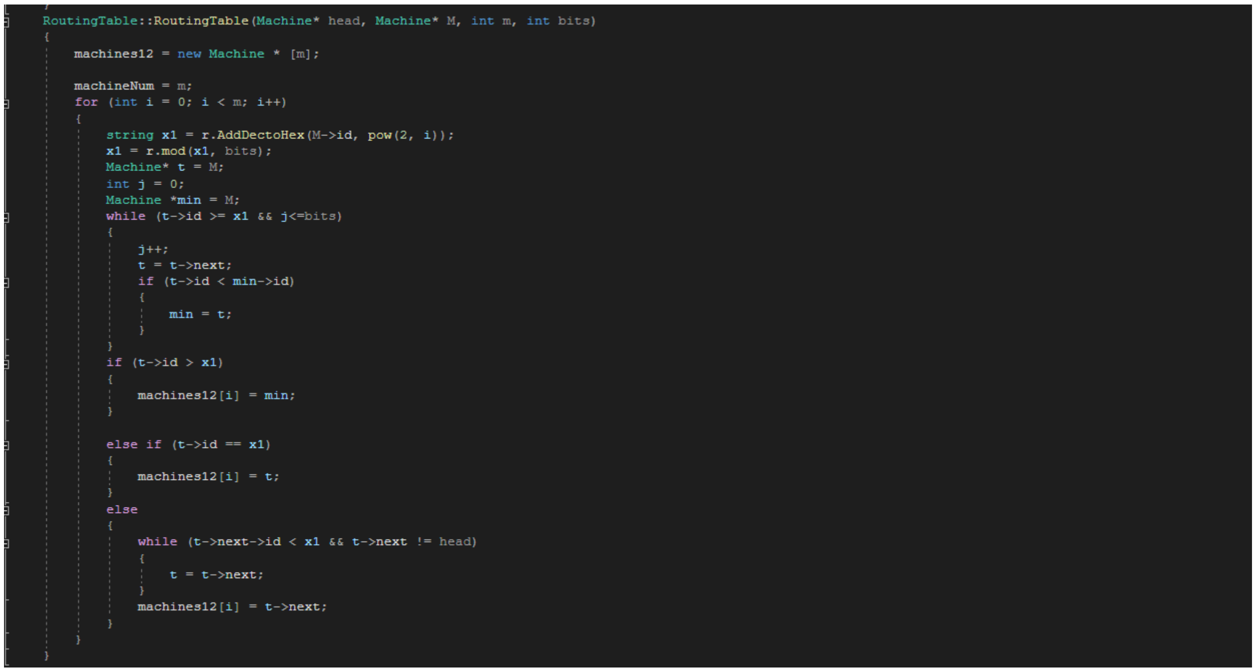
- The User is asked if they want to manually assign the ids to the machines or if they want the program to automatically assign the ids to the machines.
- For the automatic assigning of the machines a function generates random hash from a string (randomly populated when running the constructor) and then extracts a hex number in the range of bits specified by the user. A loop also checks if the machine id created is already not in

the list. Then the program assigns those ids in ascending order to the circular linked list.



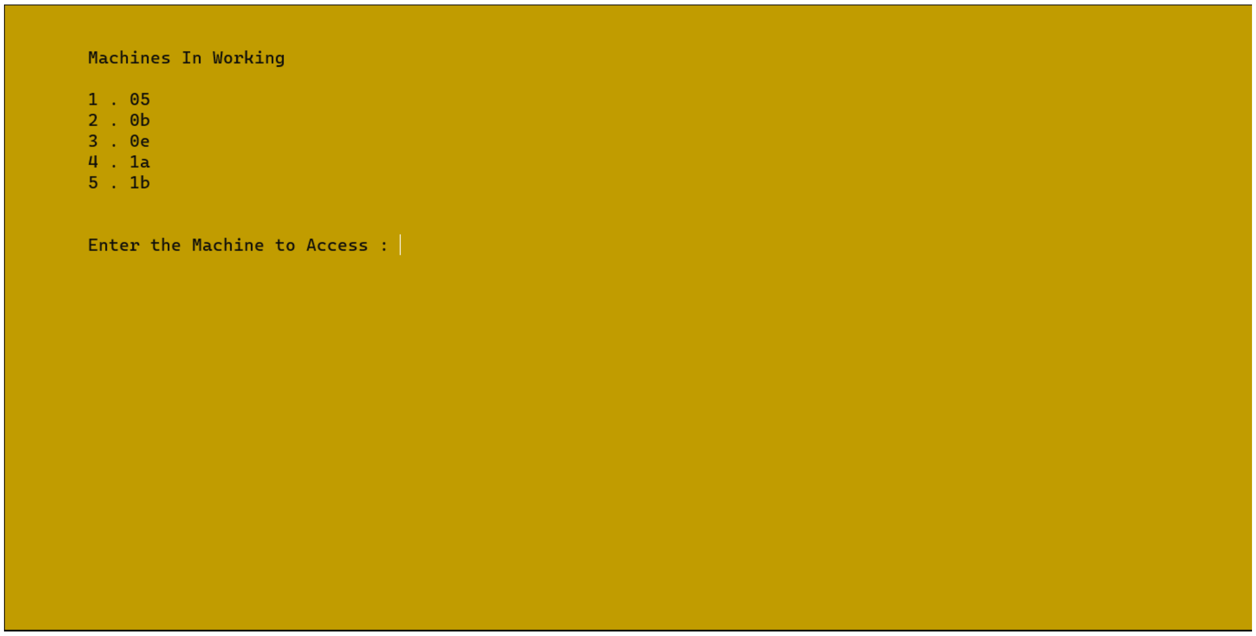
```
void DHT::insert(string d)
{
    Machine* temp = new Machine(head, d, this->treeWays, numMachine, bits);

    if (!head)
    {
        head = temp;
        head->next = head;
    }
    else if (head && head == head->next) {
        temp->next = head;
        head->next = temp;
        if (temp->id < head->id)
            head = temp;
    }
    else {
        Machine* ptr = head->next;
        Machine* prev = head;

        while (ptr != head)
        {
            prev = ptr;
            ptr = ptr->next;
        }

        if (ptr->id > d)
        {
            temp->next = ptr;
            prev->next = temp;
            head = temp;
        }
        else if (prev->id < d)
        {
            temp->next = ptr;
            prev->next = temp;
        }
        else {
            while (ptr->id < d)
            {
                prev = ptr;
                ptr = ptr->next;
            }
            temp->next = ptr;
            prev->next = temp;
        }

    }
}
```

- After the insertion of the id, a function named updateRoutingTable () is called that creates a routing table for each machine.

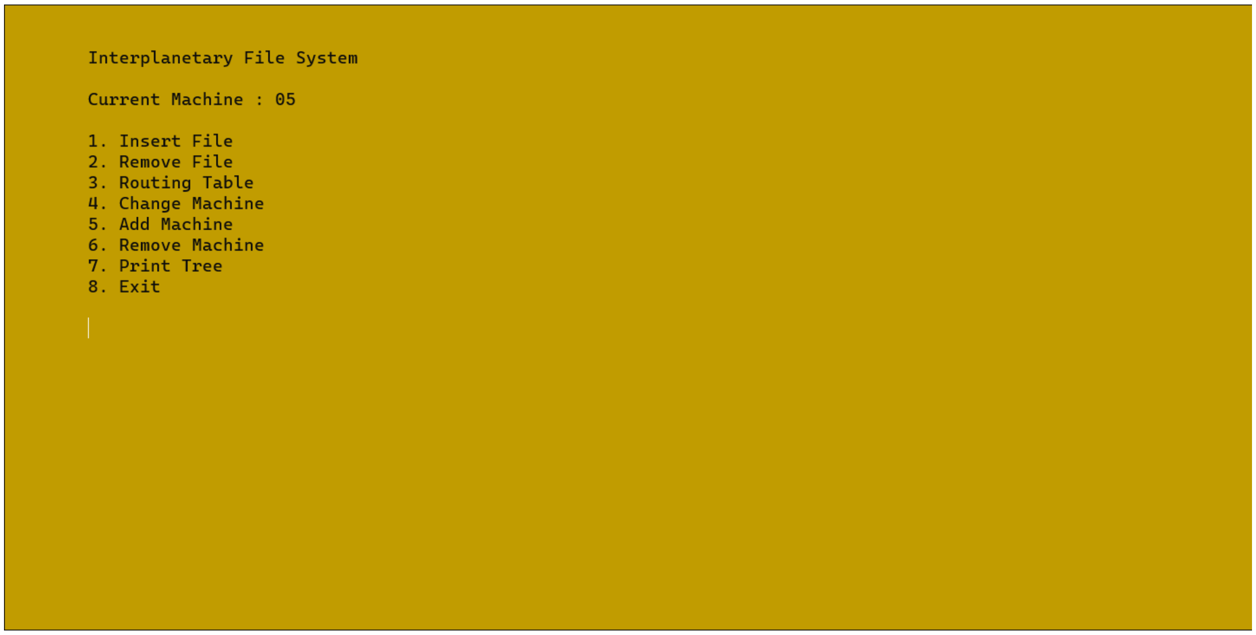


```
RoutingTable::RoutingTable(Machine* head, Machine* M, int m, int bits)
{
    machines12 = new Machine * [m];

    machineNum = m;
    for (int i = 0; i < m; i++)
    {
        string x1 = r.AddDectoHex(M->id, pow(2, i));
        x1 = r.mod(x1, bits);
        Machine* t = M;
        int j = 0;
        Machine *min = M;
        while (t->id >= x1 && j<=bits)
        {
            j++;
            t = t->next;
            if (t->id < min->id)
            {
                min = t;
            }
        }
        if (t->id > x1)
        {
            machines12[i] = min;
        }

        else if (t->id == x1)
        {
            machines12[i] = t;
        }
        else
        {
            while (t->next->id < x1 && t->next != head)
            {
                t = t->next;
            }
            machines12[i] = t->next;
        }
    }
}
```

Accessing the Machines:

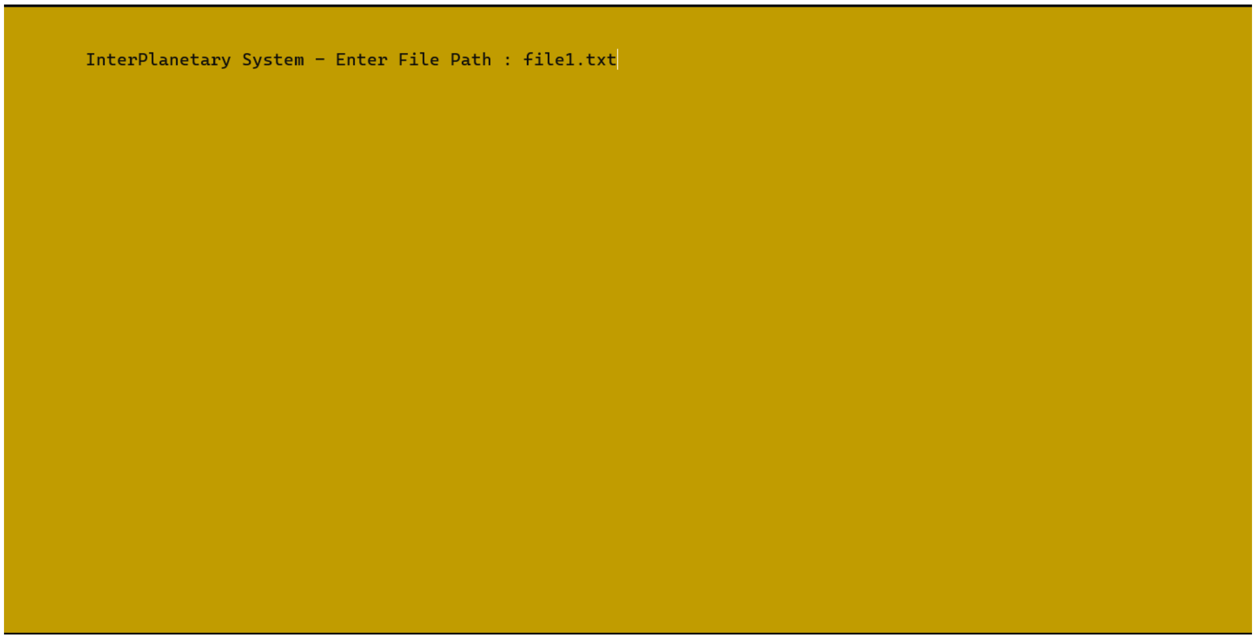

```
Machines In Working

1 . 05
2 . 0b
3 . 0e
4 . 1a
5 . 1b


Enter the Machine to Access :
```

- Then a new screen pops up that displays the current working machines and asks the user which machine they want to access. The user must input the ID of the machine to access it.


```
Interplanetary File System

Current Machine : 05

1. Insert File
2. Remove File
3. Routing Table
4. Change Machine
5. Add Machine
6. Remove Machine
7. Print Tree
8. Exit
```

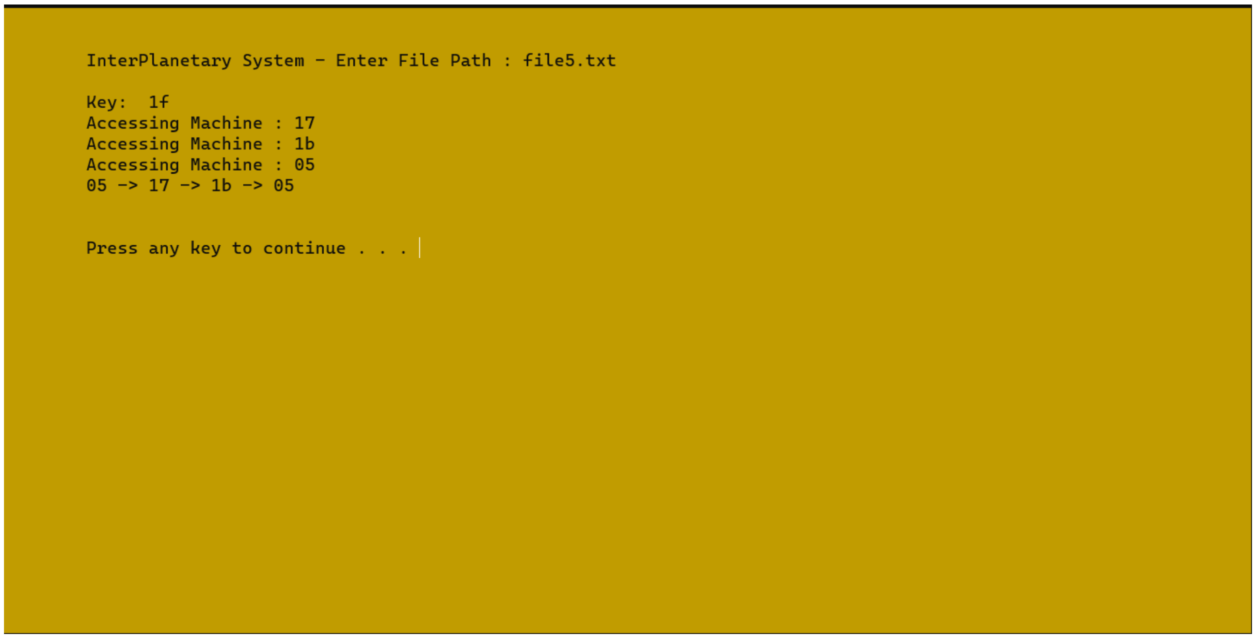
- The user is asked to input the operations they want to perform on the current machine. This is the machine from where the insertion or the deletion will start and then the current machine will change to the machine where the file was inserted or deleted from

Insertion of the File:



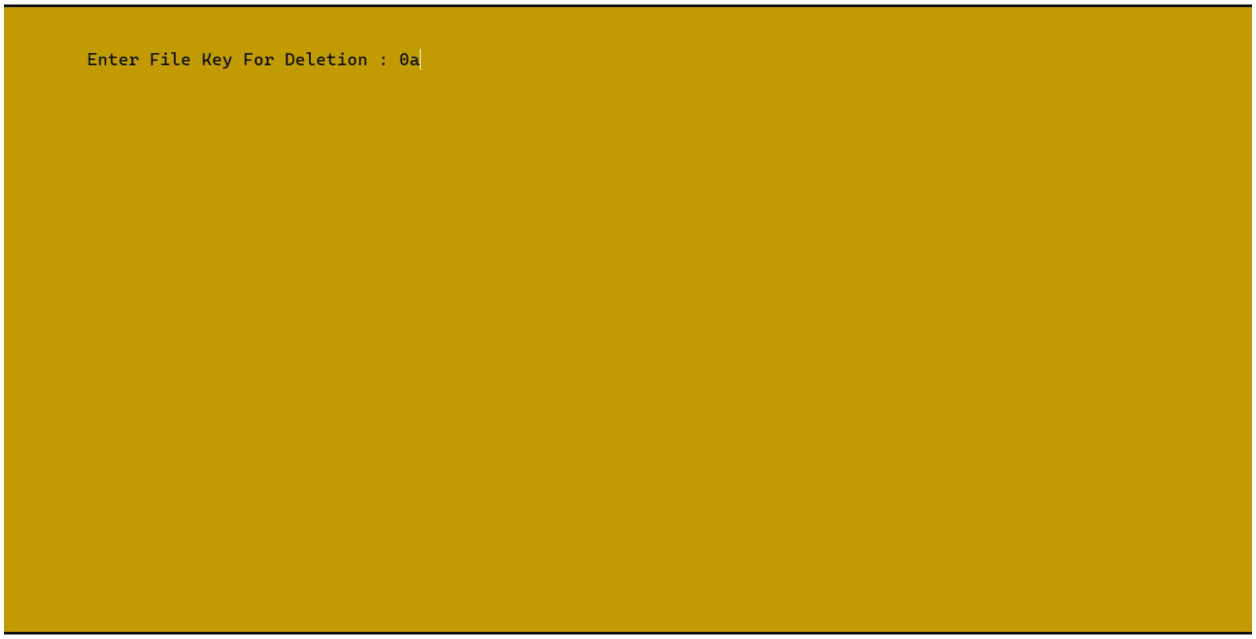
```
InterPlanetary System - Enter File Path : file1.txt
```

- The user is asked to input the path of the file they want to insert. Then SHA1 function is used to generate a hash after reading the data of the file. Then that hash is converted to a string after accessing only the desired bits of that string. Then a function Succ() is called which traverses through the routing table of each machine and finds the machine where the insertion is to take place. The key and the path of the file is passed to another function that inserts these values into the B tree of the selected machine. In the process the path taken by the machines is stored in a string and then displayed after the insertion of the file. The current machine is changed to the machine in which the file was inserted.



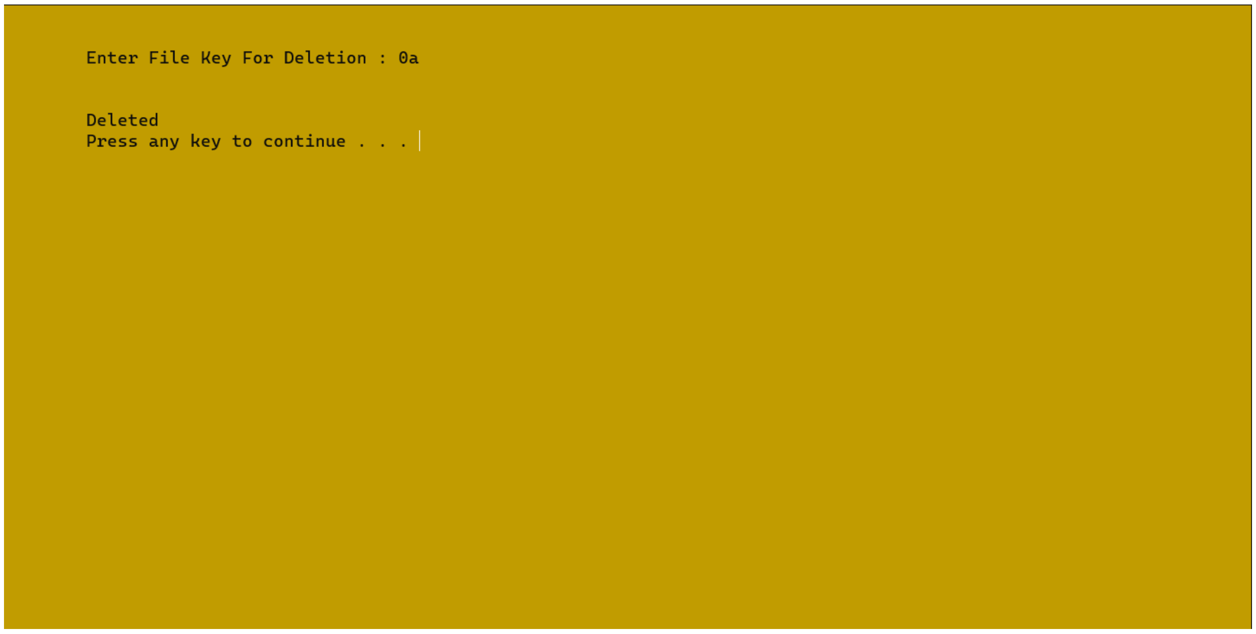
```
InterPlanetary System - Enter File Path : file5.txt

Key:  1f
Accessing Machine : 17
Accessing Machine : 1b
Accessing Machine : 05
05 -> 17 -> 1b -> 05


Press any key to continue . . .
```

Deletion of File:



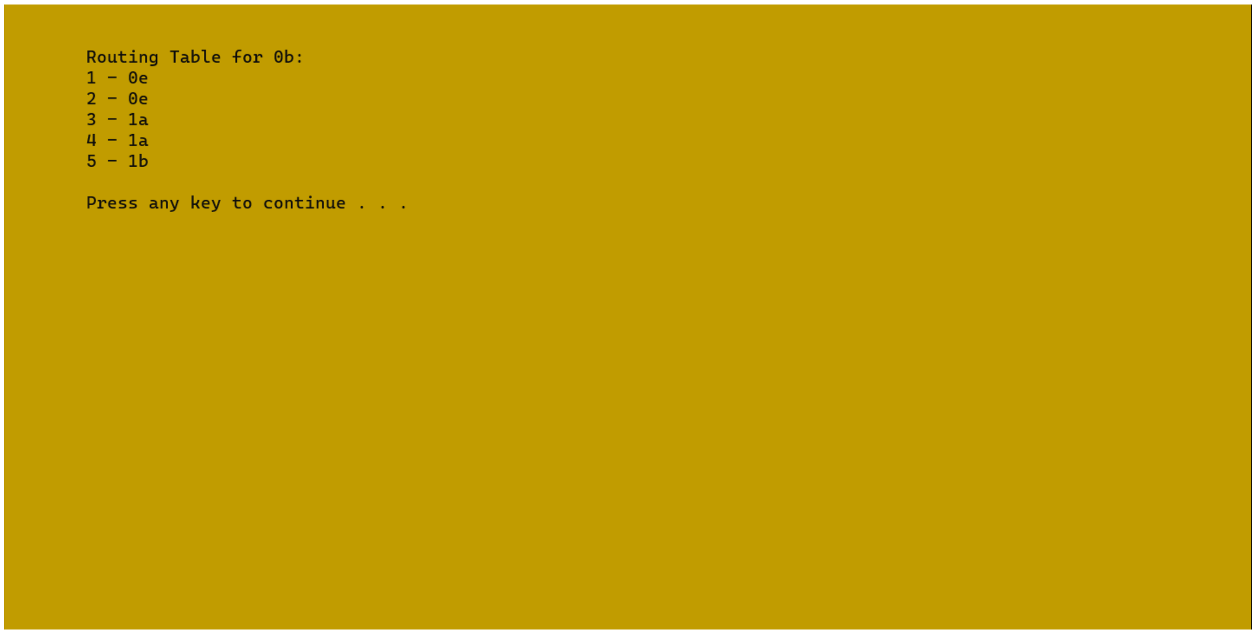
```
Enter File Key For Deletion : 0a
```

For the deletion of the file the user has to input the key of the file that they may wish to delete. First, the current machine checks its B tree for the presence of the file. If the file is not found, the Succ() function is used to find the machine where the file might be present and then remove the file form the B tree of that machine. Again, the current machine changes to the machine from where the file was deleted and the full path taken is displayed.



```
Enter File Key For Deletion : 0a


Deleted
Press any key to continue . . .
```

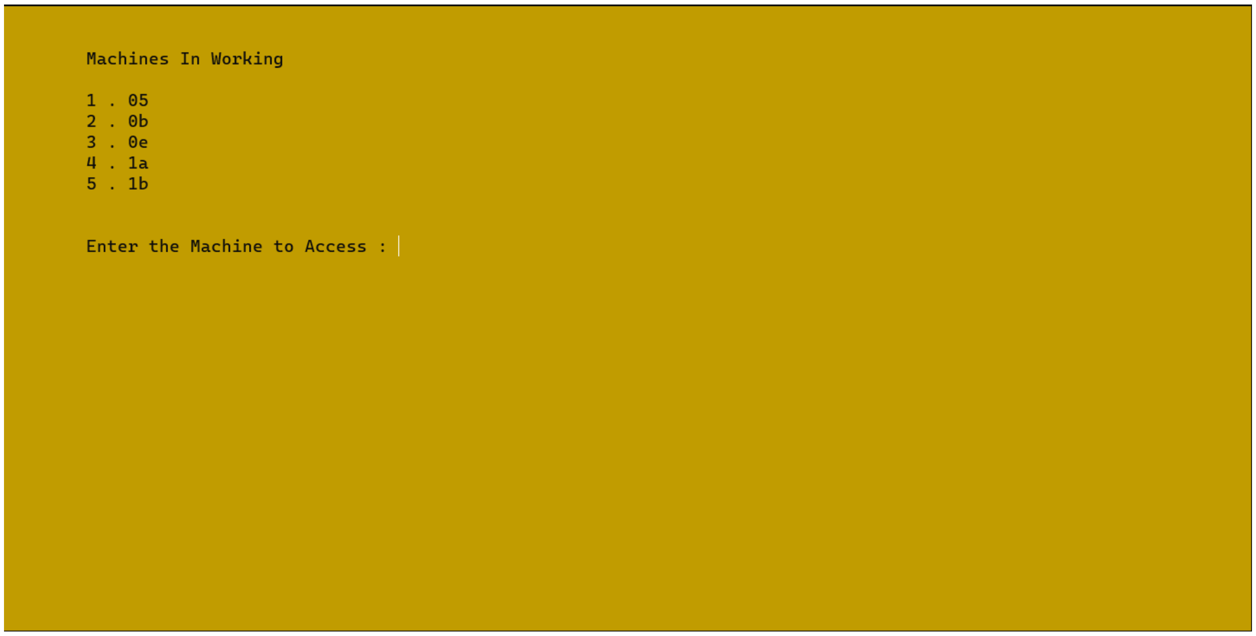
Display Routing Table:



```
Routing Table for 0b:
1 - 0e
2 - 0e
3 - 1a
4 - 1a
5 - 1b

Press any key to continue . . .
```

- The routing table of the specified machine displays the machines it is linked to and this can be used to jump to desired machines when inserting and deleting the files which reduces the time complexity of the search of desired machine to O(log n).

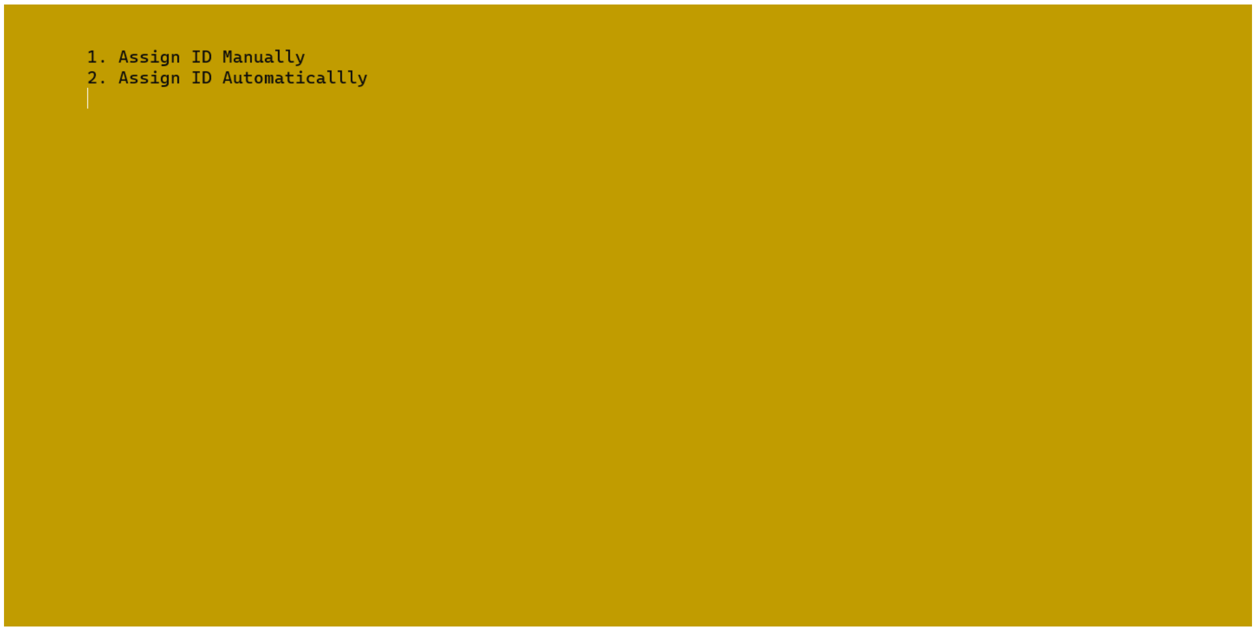
Change current machine:



```
Machines In Working

1 . 05
2 . 0b
3 . 0e
4 . 1a
5 . 1b


Enter the Machine to Access :
```

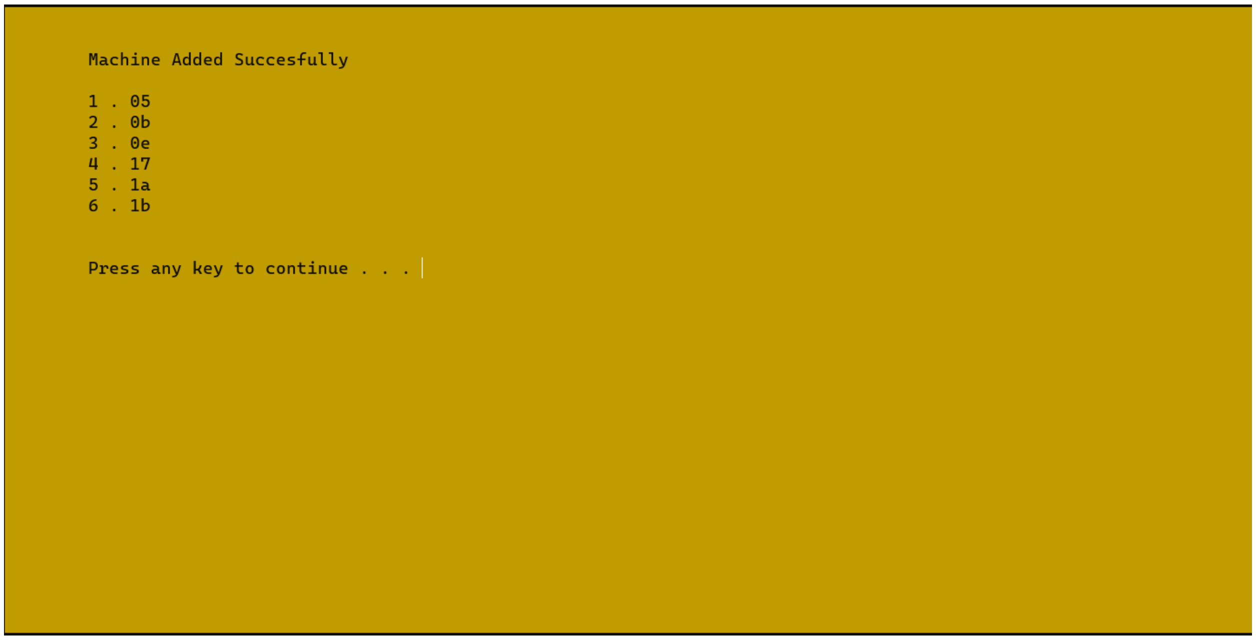
- This Options helps to change the machine from which we access the data.

Add Machine:



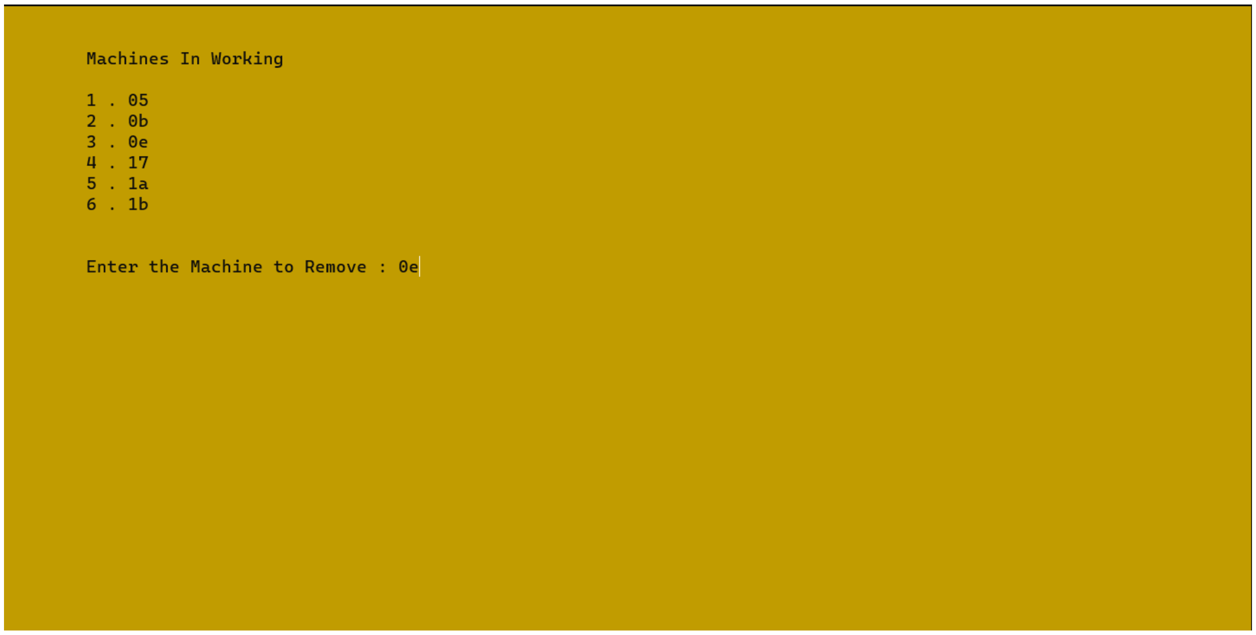
```
1. Assign ID Manually
2. Assign ID Automaticallly
```

- When the add machine option is selected, the user is asked if they want to manually assign the Id of the machine, or they want the program to automatically assign the id to the new machine. The Insert function of DHT class is called that updates the DHT by adding the new machine to system. Then again, the updateRoutingTable () function is called that updates the routing table of each of the machine. The files are then copied from the next machine that belong to the new added machine and are then added to the B tree of the new machine and removed from the next machine which helps maintain the DHT functionality.



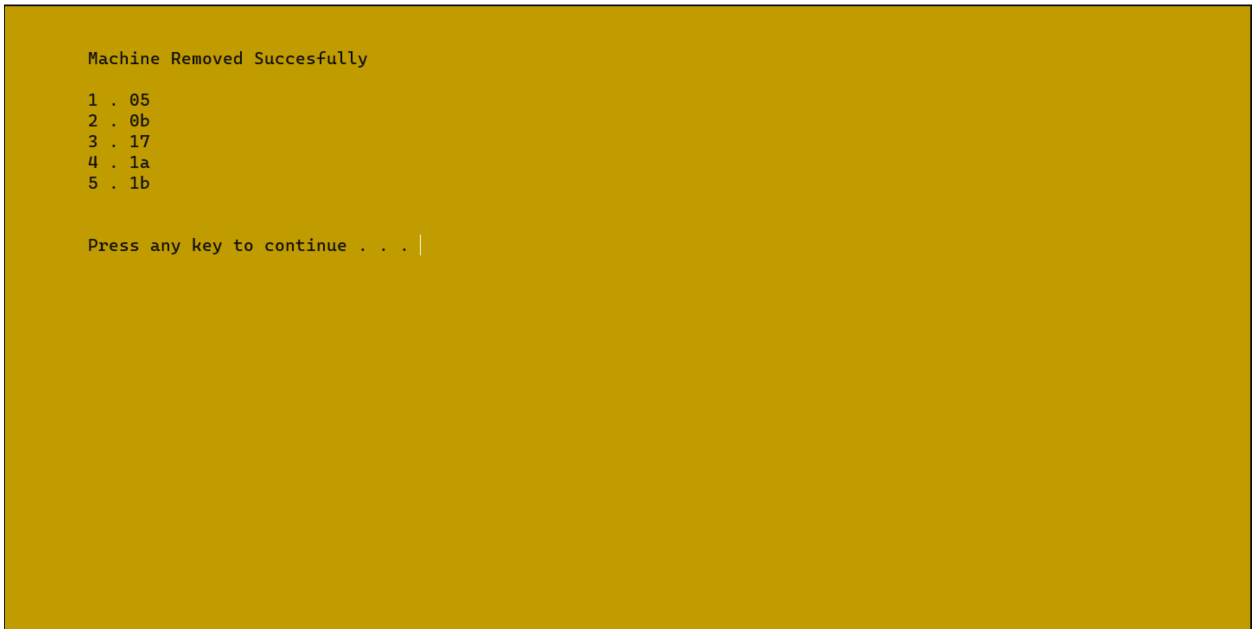
```
Machine Added Succesfully

1 . 05
2 . 0b
3 . 0e
4 . 17
5 . 1a
6 . 1b


Press any key to continue . . .
```

Remove Machine:



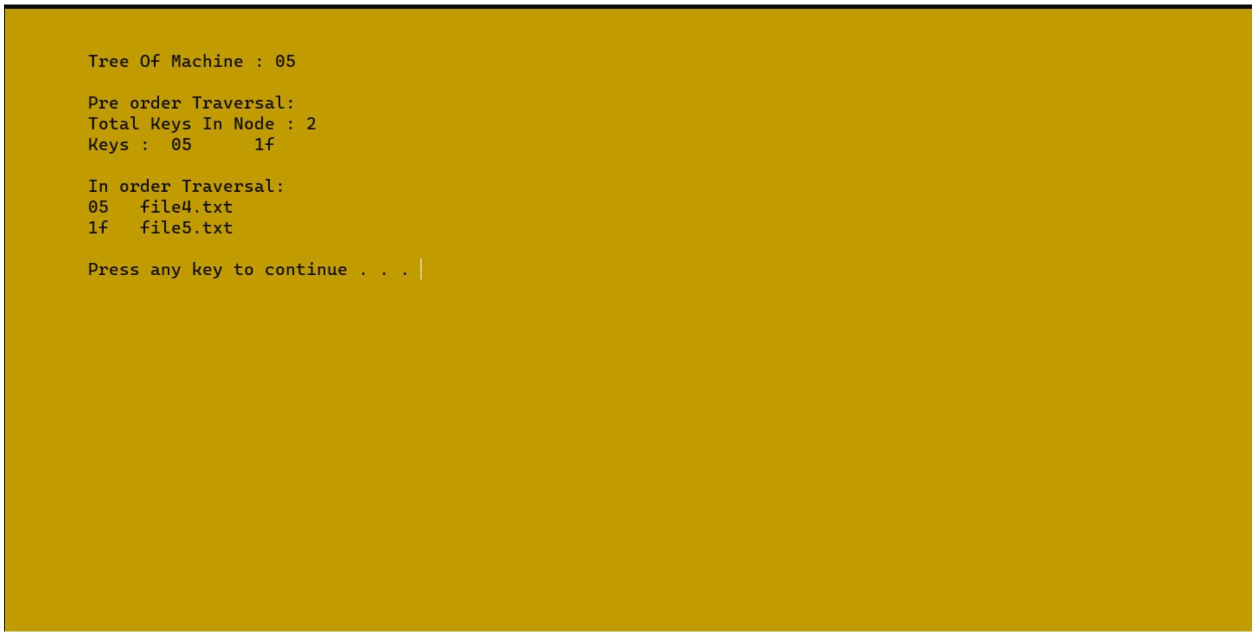
```
Machines In Working

1 . 05
2 . 0b
3 . 0e
4 . 17
5 . 1a
6 . 1b


Enter the Machine to Remove : 0e
```

- When the user selects to delete a machine, they are asked for the id of the machine they want to delete. Then the program first shifts all the files of the machine that is being deleted to the next machine's B tree and then the machine is removed from the System. Previous and the next node of that machine are relinked and remaining machines are shown



```
Machine Removed Succesfully

1 . 05
2 . 0b
3 . 17
4 . 1a
5 . 1b


Press any key to continue . . .
```

Display B tree:



```
Tree Of Machine : 05

Pre order Traversal:
Total Keys In Node : 2
Keys :  05      1f

In order Traversal:
05   file4.txt
1f   file5.txt

Press any key to continue . . .
```

- Displays the files stored in the B tree of the current machine.

Work Division:

Bilal Akbar was assigned the task to manage the whole project. He had the job of compiling all the classes and functions into a system. He created and managed the Interplanetary System class and the Distributed hash table class and functions. He also had the job to insert codes written by another member into main single project and made sure the whole program worked flawlessly. The Overall display of the file system was also programmed by Bilal Akbar.

Handling the generation of the hash through the use of the hash function was assigned to Afnan. He created function in the SHA1 class that generated hash after passing a file to them and a function that create a hash after function taking a string as input. Moreover, the hexadecimal number were used for the assigning of the ids and creation of the keys for the files so for that the functions required to manipulate those hex number were created by him. Moreover, the creation and management of the Routing table was done by Afnan.

Mehboob contributed to the project by implementing the core concepts of B-trees for the file insertion and file deletion from the system. He had work with Afnan to make sure that the files were inserted in the suitable machine and for the removal, machine was identified using the routing table making sure the time complexity does not exceed O (log n). He also assisted Bilal while addition and removal of machines as transfer of the files was necessary between the B trees.

The group had to work in close collaboration to ensure that the program had no error and it fulfilled its desired purpose. Overall, we had a great time together learning new concepts of data structures.