Date 25/11/2020

Expt. No. 8 Page No. 1

Functions of Dictionary using Hashing

```
typedef struct list {
    int data;
    struct list *next;
} node_type;
node_type *ptr[max], *root[max], *temp[max];

class Dictionary {
public:
    int index;
    Dictionary();
    void insert(int);
    void search(int);
    void delete_ele(int);
};
Dictionary::Dictionary()
{
    index = -1;
    for (int i=0; i<max; i++)
    {
        root[i] = NULL;
        ptr[i] = NULL;
        temp[i] = NULL;
    }
}
void Dictionary::insert(int key) {
    index = int(key % max);
    ptr[index] = (node_type*) malloc(sizeof(node_type));
    ptr[index]->data = key;
```

```
if (root[index]==NULL){
    root[index]= ptr[index];
    root[index]->next=NULL;
    temp[index] = ptr[index]; }
else {
    temp[index]=root[index];
    while (temp[index]->next!= NULL)
        temp[index] = temp[index]->next;
    temp[index]->next= ptr[index]; } }

void Dictionary:: search (int key) {
    int flag=0;
    index = int (key %.max);
    temp[index] = root[index];
    while (temp[index]!=NULL){
        if (temp[index]->data==key ) {
            cout << "\n Key found";
            flag=1;
            breake;
        }else temp[index] = temp[index]->next;
    } if (flag==0) cout<<"\n Key not found"; }
```