Implementing dictonary using hash

```
class node
{   string key, value;
    friend class dictionary;
    node ()
    {   next= NULL;
    }
    node (string key, string value)
    {       this->key =key;
            this ->value=value;
            next =NULL;
    }
};
class dictionary
{
      node head[MAX];
      public:
      dictionary ()
      {   for (int  i=0; i <Max ; i++)
          {      head[i]=NULL;
      }
      int hashf (string  word)
      bool insert (string, string)
      string find (string word);
      bool deleteword (string word)
}
```

```
String find (string word)
{   int index = hashfunction (word);
    node start = head [index];
    if (start == NULL)
        return "-1";
    while (start != NULL)
    {   if (start ->key == word)
            return start ->value;
            start = start -> next;
    }
    return "-1";
}
bool insert(string word, string meaning)
{   int index = hashfunction (word);
    node p = new node(word, meaning);
    if (head[index] == NULL)
    {   head[index] = p.
        cout << "\n" << word << "inserted"
    }   return true;
    else
    {   node start = head {index];
        while (start-> next != NULL)
            start = start -> next;
    }   start -> next = p;
        return true;
    return false;
}
```

```
int hash_func (string word)
{
        int sum = 0;
      { for (i=0; i< word.length(); i++)
                sum = sum + word [i];
        }
        return (sum % 100)
}
```