```
Dictonary( )
{
index → i = -1, j = 0;
  while ( j < max )
  {
      root, ptr, tmp = NULL ;
  } j++;
  }
}

         data
insert( )
{
    i = data % max ;
   ptr[i] → val = data ;
   if ( root [i] == NULL )
   {
      root [i] = temp[i] = ptr[i] ;
      root [i] → nxt = NULL
   }
   else
   {
     // insert ptr[i] at the end of
     // temp
     while ( temp → nxt != NULL )
             temp = temp → nxt ;
     tmp → nxt = ptr[i]
}  }
```

```
delete (data)
{
    i = data % max;
    tmp[i] = root[i];
    While ( tmp[i] -> val != data and tmp[i] != NULL)
    {
        *ptr = *temp;
        temp = temp -> nxt;
    }
    ptr[i] -> nxt = tmp[i] -> nxt;
    temp[i] -> val = -1;
    tmp[i] = NULL;
    free(tmp[i]);
}

struct hash        // Initialize
{
    int val;
    struct hash *nxt;
} [illegible];
typedef struct hash node;
node *tmp[max], *root[max], *ptr[max];
int index;
```