```
class Entry_Hashtable
  {
     public:
      int k, v;
       Entry_Hashtable (int k, int v)
              {
                this -> k = k
                this -> v = v;
            }
        };
class Map_Hashtable
  {
    private: Entry_Hashtable **t;
    public:
       • Map Hashtable(){
       t = new Entry_Hashtable *[size];
         for (i=0; i< size; i++)
              t[i] = NULL;
       }
  int Hashfunc (int k)
       {
       return k%size;
  }


void insert (int k, int v)
   {
     int h= Hashfunc(k);
while (t[h] != NULL && t[h] -> k != k)
    {
        h = Hashfunc(h+1); // linear probing
    }

    if (t[h] != NULL)
        delete t[h];
    t[h] = new Entry_Hashtable(k, v);
}
int delete (int k)
{
    int h = Hashfunc(k);
    while (t[h] != NULL)
     {
        if (t[h] -> k == k)
            break;
        h = HashFunc(h+1);
     }
    if (t[h] == NULL) {
        cout << "no element found".
        return;
     }
    else {
        delete (t[h]);
        cout << "Element deleted";
     }
  }
};
```