Chaining

```
#ifndef Chains_h
#define Chains_h
#include<stdlib.h>
struct Node
{
    int data;
    struct Node *next;
};

void SortedInsert(struct Node **H,int x)
{
    struct Node *t,*q=NULL,*p=*H;

    t=(struct Node*)malloc(sizeof(struct Node));
    t->data=x;
    t->next=NULL;

    if(*H==NULL)
        *H=t;
    else
    {
        while(p && p->data<x)
        {
            q=p;
            p=p->next;
        }
        if(p==*H)
        {
            t->next=*H;
            *H=t;
        }
        else
        {
            t->next=q->next;
            q->next=t;
        }
    }

}
struct Node *Search(struct Node *p,int key)
{
    while(p!=NULL)
    {
        if(key==p->data)
```

```
        {
            return p;
        }
        p=p->next;
    }
    return NULL;

}


#endif /* Chains_h */



#include <stdio.h>
#include "Chains.h"

int hash(int key)
{
    return key%10;
}

void Insert(struct Node *H[],int key)
{
    int index=hash(key);
    SortedInsert(&H[index],key);
}

int main()
{
    struct Node *HT[10];
    struct Node *temp;
    int i;

    for(i=0;i<10;i++)
        HT[i]=NULL;

    Insert(HT,12);
    Insert(HT,22);
    Insert(HT,42);

    temp=Search(HT[hash(21)],21);

    printf("%d ",temp->data);


    return 0;
}
```