```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define p 11
#define HMax 11
#define VMax 100
#define EndOfList -1

typedef struct{
 double paid;
 int service;
} visit_type;

typedef visit_type ListElementType;
typedef struct {



    int RecKey;
    ListElementType Data;
    int Link;
} ListElm;

typedef struct  {
    int HashTable[HMax];
    int Size;
    int SubListPtr;
    int StackPtr;
    ListElm List[VMax];
} HashListType;

typedef enum {
    FALSE, TRUE
} boolean;

void CreateHashList(HashListType *HList);
int HashKey(int Key);
boolean FullHashList(HashListType HList);
void SearchSynonymList(HashListType HList,int KeyArg,int *Loc,int *Pred);
void SearchHashList(HashListType HList,int KeyArg,int *Loc,int *Pred);
void AddRec(HashListType *HList,ListElm InRec);
void DeleteRec(HashListType *HList,int DelKey);

void menu(int *choise);
void Client_insert(HashListType *HList);
int name_conversion(char name[]);
void Search_client(HashListType HList);

int main()
{
    int choise;
    HashListType HList;

    while(!FullHashList(HList))
    {
        menu(&choise);

        switch(choise)
        {
            case 1:
                CreateHashList(&HList);
                break;
            case 2:
                Client_insert(&HList);
                break;
```

```
            case 3:
                Search_client(HList);
                break;
            case 4:
                return 0;
        }
    }
}

int HashKey(int Key)

{
    return Key%HMax;
}

void CreateHashList(HashListType *HList)

{
    int index;

    HList->Size=0;
    HList->StackPtr=0;


    index=0;
    while (index<HMax)
    {
        HList->HashTable[index]=EndOfList;
        index=index+1;
    }


    index=0;
    while(index < VMax-1)
    {
        HList->List[index].Link=index+1;
        HList->List[index].Data.paid=0;
        HList->List[index].Data.service=0;
        index=index+1;
    }
    HList->List[index].Link=EndOfList;

}

boolean FullHashList(HashListType HList)
{
    return(HList.Size==VMax);
}

void SearchSynonymList(HashListType HList,int KeyArg,int *Loc,int *Pred)
{
    int Next;
    Next=HList.SubListPtr;
    *Loc=-1;
    *Pred=-1;
    while(Next!=EndOfList)
    {
        if (HList.List[Next].RecKey==KeyArg)
        {
            *Loc=Next;
            Next=EndOfList;
        }
        else
        {
            *Pred=Next;
            Next=HList.List[Next].Link;
```

```
        }
    }
}
void SearchHashList(HashListType HList,int KeyArg,int *Loc,int *Pred)
{
    int HVal;
    HVal=HashKey(KeyArg);
    if (HList.HashTable[HVal]==EndOfList)
    {
        *Pred=-1;
        *Loc=-1;
    }
    else
    {
        HList.SubListPtr=HList.HashTable[HVal];
        SearchSynonymList(HList,KeyArg,Loc,Pred);
    }
}

void AddRec(HashListType *HList,ListElm InRec)
{
    int Loc, Pred, New, HVal;


    if(!(FullHashList(*HList)))
    {
        Loc=-1;
        Pred=-1;
        SearchHashList(*HList,InRec.RecKey,&Loc,&Pred);
        if(Loc==-1)
        {
            HList->Size=HList->Size +1;
            New=HList->StackPtr;
            HList->StackPtr=HList->List[New].Link;
            HList->List[New]=InRec;
            if (Pred==-1)
            {
                HVal=HashKey(InRec.RecKey);
                HList->HashTable[HVal]=New;
                HList->List[New].Link=EndOfList;
            }
            else
            {
                HList->List[New].Link=HList->List[Pred].Link;
                HList->List[Pred].Link=New;
            }
        }

        else
        {
            printf("YPARXEI HDH EGGRAFH ME TO IDIO KLEIDI \n");
        }
    }
    else
    {
        printf("Full list...");
    }
}
void DeleteRec(HashListType *HList,int DelKey)
{
    int Loc, Pred, HVal;

    SearchHashList(*HList,DelKey,&Loc,&Pred);
    if(Loc!=-1)
    {
        if(Pred!=-1)
```

```
        {
            HList->List[Pred].Link=HList->List[Loc].Link;
        }
        else
        {
            HVal=HashKey(DelKey);
            HList->HashTable[HVal]=HList->List[Loc].Link;
        }
        HList->List[Loc].Link=HList->StackPtr;
        HList->StackPtr=Loc;
        HList->Size=HList->Size -1;
    }
    else
    {
        printf("DEN YPARXEI  EGGRAFH ME KLEIDI %d \n",DelKey);
    }
}


//????????? ????????? ?????.?????????? ??? ???? ??? ??????? ? ???????,
void menu(int *choise)
{
    printf("                       MENOY                     \n");
    printf("-------------------------------------------------\n");
    printf("1. CREATE DATABASE\n");
    printf("2. INSERT APPOINTMENT\n");
    printf("3. PRINT CLIENT'S APPOINTMENTS\n");
    printf("4. EXIT\n");

    do{
        printf("\nCHOISE: ");
        scanf("%d",&(*choise));

        if(*choise < 1 || *choise > 4)
            printf("Choise must be between 1-4.Try again.\n");

    }while(*choise < 1 || *choise > 4);

}

//????????? ??? ??? ??????? 2, ??????? ?? ???????? ??? HList ???.
void Client_insert(HashListType *HList)
{
    ListElm AnItem;
    char name[20];
    char cont;

    while(!FullHashList(*HList))
    {

        printf("\nEnter the client's Name: ");
        scanf("%s",name);
        name[strlen(name)]='\0';
        AnItem.RecKey = name_conversion(name);

        printf("\nEnter the service:\n");
        printf("1-Whitening\n");
        printf("2-Cleaning\n");
        printf("3-Extraction\n");
        do{
            scanf("%d",&AnItem.Data.service);

            if(AnItem.Data.service < 1 || AnItem.Data.service > 3)
                printf("Service choise must be between 1-3.Try Again.\n");
        }while(AnItem.Data.service < 1 || AnItem.Data.service > 3);

```

```
265         printf("\nEnter the amount paid: ");
266         scanf("%lf",&AnItem.Data.paid);
267
268         AnItem.Link = EndOfList;
269
270         printf("\nContinue Y/N:");
271         scanf(" %c",&cont);
272
273         AddRec(&(*HList),AnItem);
274
275         if(cont == 'N')
276             break;
277
278     }
279 }
280
281 //????????? ??? ???????? ??? ?????????? ??? ?????????? ?? ???? ???????????????(???? ??? ??????? ????
????????).
282 int name_conversion(char name[])
283 {
284     int sum = 0,i;
285
286     for(i = 0; name[i] != '\0'; i++)
287         sum += (i+1) * (name[i] - 64);
288
289     return sum % p;
290
291 }
292
293 //????????? ?????????? ?? ???? ?? ????? ,?? ?????? ? ??????? ????????? ?? ???????? ???,?????? ??????????
??????.
294 void Search_client(HashListType HList)
295 {
296     int Loc,Pred,key;
297     char name[20];
298
299     printf("\nEnter the client's Name: ");
300     scanf("%s",name);
301     name[strlen(name)]='\0';
302     key = name_conversion(name);
303
304     SearchHashList(HList,key,&Loc,&Pred);
305
306     if(Loc != -1)
307     {
308         printf("Service: %d\n",HList.List[Loc].Data.service);
309         printf("Amount Paid: %.2lf\n",HList.List[Loc].Data.paid);
310     }
311     else
312         printf("DEN YPARXEI TETOIA EGGRAFH");
313 }
314
315
316
```