

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define p 11
6  #define HMax 11
7  #define VMax 100
8  #define EndOfList -1
9
10 typedef struct{
11     double paid;
12     int service;
13 } visit_type;
14
15 typedef visit_type ListElementType;
16 typedef struct {
17
18
19
20     int RecKey;
21     ListElementType Data;
22     int Link;
23 } ListElm;
24
25 typedef struct {
26     int HashTable[HMax];
27     int Size;
28     int SubListPtr;
29     int StackPtr;
30     ListElm List[VMax];
31 } HashListType;
32
33 typedef enum {
34     FALSE, TRUE
35 } boolean;
36
37 void CreateHashList(HashListType *HList);
38 int HashKey(int Key);
39 boolean FullHashList(HashListType HList);
40 void SearchSynonymList(HashListType HList,int KeyArg,int *Loc,int *Pred);
41 void SearchHashList(HashListType HList,int KeyArg,int *Loc,int *Pred);
42 void AddRec(HashListType *HList,ListElm InRec);
43 void DeleteRec(HashListType *HList,int DelKey);
44
45 void menu(int *choise);
46 void Client_insert(HashListType *HList);
47 int name_conversion(char name[]);
48 void Search_client(HashListType HList);
49
50 int main()
51 {
52     int choise;
53     HashListType HList;
54
55     while(!FullHashList(HList))
56     {
57         menu(&choise);
58
59         switch(choise)
60         {
61             case 1:
62                 CreateHashList(&HList);
63                 break;
64             case 2:
65                 Client_insert(&HList);
66                 break;

```

```

67         case 3:
68             Search_client(HList);
69             break;
70         case 4:
71             return 0;
72     }
73 }
74 }
75
76 int HashKey(int Key)
77 {
78     return Key%HMax;
79 }
80
81 void CreateHashList(HashListType *HList)
82 {
83     int index;
84
85     HList->Size=0;
86     HList->StackPtr=0;
87
88     index=0;
89     while (index<HMax)
90     {
91         HList->HashTable[index]=EndOfList;
92         index=index+1;
93     }
94
95     index=0;
96     while(index < VMax-1)
97     {
98         HList->List[index].Link=index+1;
99         HList->List[index].Data.paid=0;
100        HList->List[index].Data.service=0;
101        index=index+1;
102    }
103    HList->List[index].Link=EndOfList;
104 }
105
106 boolean FullHashList(HashListType HList)
107 {
108     return(HList.Size==VMax);
109 }
110
111 void SearchSynonymList(HashListType HList,int KeyArg,int *Loc,int *Pred)
112 {
113     int Next;
114     Next=HList.SubListPtr;
115     *Loc=-1;
116     *Pred=-1;
117     while(Next!=EndOfList)
118     {
119         if (HList.List[Next].RecKey==KeyArg)
120         {
121             *Loc=Next;
122             Next=EndOfList;
123         }
124         else
125         {
126             *Pred=Next;
127             Next=HList.List[Next].Link;
128         }
129     }
130 }

```

```

133     }
134 }
135 }
136 void SearchHashList(HashListType HList,int KeyArg,int *Loc,int *Pred)
137 {
138     int HVal;
139     HVal=HashKey(KeyArg);
140     if (HList.HashTable[HVal]==EndOfList)
141     {
142         *Pred=-1;
143         *Loc=-1;
144     }
145     else
146     {
147         HList.SubListPtr=HList.HashTable[HVal];
148         SearchSynonymList(HList,KeyArg,Loc,Pred);
149     }
150 }
151
152 void AddRec(HashListType *HList,ListElm InRec)
153 {
154     int Loc, Pred, New, HVal;
155
156
157     if(!(FullHashList(*HList)))
158     {
159         Loc=-1;
160         Pred=-1;
161         SearchHashList(*HList,InRec.RecKey,&Loc,&Pred);
162         if(Loc===-1)
163         {
164             HList->Size=HList->Size +1;
165             New=HList->StackPtr;
166             HList->StackPtr=HList->List[New].Link;
167             HList->List[New]=InRec;
168             if (Pred===-1)
169             {
170                 HVal=HashKey(InRec.RecKey);
171                 HList->HashTable[HVal]=New;
172                 HList->List[New].Link=EndOfList;
173             }
174             else
175             {
176                 HList->List[New].Link=HList->List[Pred].Link;
177                 HList->List[Pred].Link=New;
178             }
179         }
180
181         else
182         {
183             printf("YPARXEI HDH EGGRAPH ME TO IDIO KLEIDI \n");
184         }
185     }
186     else
187     {
188         printf("Full list...");
189     }
190 }
191 void DeleteRec(HashListType *HList,int DelKey)
192 {
193     int Loc, Pred, HVal;
194
195     SearchHashList(*HList,DelKey,&Loc,&Pred);
196     if(Loc!=-1)
197     {
198         if(Pred!=-1)

```

```

199     {
200         HList->List[Pred].Link=HList->List[Loc].Link;
201     }
202     else
203     {
204         HVal=HashKey(DelKey);
205         HList->HashTable[HVal]=HList->List[Loc].Link;
206     }
207     HList->List[Loc].Link=HList->StackPtr;
208     HList->StackPtr=Loc;
209     HList->Size=HList->Size -1;
210 }
211 else
212 {
213     printf("DEN YPARXEI EGGRAPH ME KLEIDI %d \n",DelKey);
214 }
215 }
216
217
218 //????????? ?????????? ?????????????? ??? ??? ???? ??????? ? ???????,
219 void menu(int *choise)
220 {
221     printf("                      MENOY                      \n");
222     printf("-----\n");
223     printf("1. CREATE DATABASE\n");
224     printf("2. INSERT APPOINTMENT\n");
225     printf("3. PRINT CLIENT'S APPOINTMENTS\n");
226     printf("4. EXIT\n");
227
228     do{
229         printf("\nCHOISE: ");
230         scanf("%d",&(*choise));
231
232         if(*choise < 1 || *choise > 4)
233             printf("Choise must be between 1-4.Try again.\n");
234
235     }while(*choise < 1 || *choise > 4);
236
237 }
238
239 //????????? ??? ??? ??????? 2, ??????? ?? ????????? ??? HList ???
240 void Client_insert(HashListType *HList)
241 {
242     ListElm AnItem;
243     char name[20];
244     char cont;
245
246     while(!FullHashList(*HList))
247     {
248
249         printf("\nEnter the client's Name: ");
250         scanf("%s",name);
251         name[strlen(name)]='\0';
252         AnItem.RecKey = name_conversion(name);
253
254         printf("\nEnter the service:\n");
255         printf("1-Whitening\n");
256         printf("2-Cleaning\n");
257         printf("3-Extraction\n");
258         do{
259             scanf("%d",&AnItem.Data.service);
260
261             if(AnItem.Data.service < 1 || AnItem.Data.service > 3)
262                 printf("Service choise must be between 1-3.Try Again.\n");
263         }while(AnItem.Data.service < 1 || AnItem.Data.service > 3);
264

```

```

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

```