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

#define HMax 10

#define VMax 100
#define EndOfList -1

typedef char ListElementType;

typedef struct {
    char RecKey[8];
    ListElementType Password[6];
    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(char Key[]);
boolean FullHashList(HashListType HList);
void SearchSynonymList(HashListType HList,char KeyArg[],int *Loc,int *Pred);
void SearchHashList(HashListType HList,char KeyArg[],int *Loc,int *Pred);
void AddRec(HashListType *HList,ListElm InRec);
void DeleteRec(HashListType *HList,char DelKey[]);


int findAverage(char username[]);
void BuildHashList(HashListType *HList);
/*????? ????? ?? ??????????? ??????? ???? ?????? ????????? ??? ???????? ????????????? ???? ???????????
(SearchSynonymList , SearchHashList, CreateHashList)*/
int main()
{
    //?????? ??????????.
    HashListType HList;
    ListElm AnItem;
    int Loc,Pred;
    char newEntry;

    //????? ?????????? ??? ?????????? HashList.
    BuildHashList(&HList);

    //???????????? ???? ????????? ????????? ????? ?? ??? ?????????? ? ???????.
    while(TRUE)
    {
        //???????? ????????? USERNAME ??? PASSWORD.
        printf("USERNAME: ");
        scanf("%s",AnItem.RecKey);
        AnItem.RecKey[strlen(AnItem.RecKey)] = '\0';
        printf("PASSWORD: ");
        scanf("%s",AnItem.Password);
        AnItem.Password[strlen(AnItem.Password)] = '\0';

        //???????? ??? ????????? ??? ???????.
```

```
 66          SearchHashList(HList,AnItem.RecKey,&Loc,&Pred);
 67
 68          //?? ??????? ?? USERNAME.
 69          if( Loc != -1)
 70              //??????? ??? PASSWORD ??? ??????????? ??? USERNAME ??? ??????? , ?? ?? PASSWORD ??? ?????? ???
?? ???????????.
 71              //?? ????? ????.
 72              if(strcmp(HList.List[Loc].Password,AnItem.Password) == 0)
 73                  //???????? ????????? ????????? ?????????? ??? ???????.
 74                  printf("You have logged in to the system.\n");
 75              //?? ??? ????? ????.
 76              else
 77                  //???????? ????????? ?????????? ?????????? ??? ??????? ??? ??? ? ??????? ?? ?????????.
 78                  printf("Access is forbidden: Wrong password.\n");
 79          //?? ??? ??????? ?? USERNAME.
 80          else
 81              //??????????? ??????.
 82              printf("Access is forbidden: Wrong user ID.\n");
 83
 84          //?????? ?? ????? ?? ????????? ? ??????? ?? ??????? ??? ???? ????????.
 85          printf("New entry Y/N (Y=Yes, N=No)?");
 86
 87          do{
 88              scanf(" %c", &newEntry);
 89
 90              if(newEntry != 'Y' && newEntry != 'N')
 91                  printf("The answer must be Y(Yes) or N(No).Try again:");
 92
 93          }while(newEntry != 'Y' && newEntry != 'N');
 94
 95          //?? ???????? '?'.????? ????????????? ???????????.
 96          if(newEntry == 'N')
 97              break;
 98      }
 99      return 0;
100  }
101
102  /*??????????? ??? ?????????? HashKey ???? ?? ????? ??????? ?? ??????????? ?????????????? ???
???????????.*/
103  int HashKey(char Key[])
104  {
105      return findAverage(Key)%HMax;
106  }
107
108  void CreateHashList(HashListType *HList)
109
110  {
111      int index;
112
113      HList->Size=0;
114      HList->StackPtr=0;
115
116      index=0;
117      while (index<HMax)
118      {
119          HList->HashTable[index]=EndOfList;
120          index=index+1;
121      }
122
123
124      index=0;
125      while(index < VMax-1)
126      {
127          HList->List[index].Link=index+1;
128          strcpy(HList->List[index].Password,"0");
129          index=index+1;
```

```
    }
    HList->List[index].Link=EndOfList;
}

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

void SearchSynonymList(HashListType HList,char KeyArg[],int *Loc,int *Pred)
{
    int Next;
    Next=HList.SubListPtr;
    *Loc=-1;
    *Pred=-1;
    while(Next!=EndOfList)
    {
        if (strcmp(HList.List[Next].RecKey,KeyArg)==0)
        {
            *Loc=Next;
            Next=EndOfList;
        }
        else
        {
            *Pred=Next;
            Next=HList.List[Next].Link;
        }
    }
}
void SearchHashList(HashListType HList,char 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,char 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 %s \n",DelKey);
    }
}

/*????????? ??????? ??? ????? average ??? ??? ????????? ??????????????.*/
int findAverage(char username[])

{   //?????? ??????????.
    //????????? ???? ????????? average ??? ?????? ASCII ??? ?????? ????????? ??? USERNAME.
    int average=username[0],i;

    for(i=0; username[i] != '\0'; i++)
        if(username[i+1] == '\0')
            //??????????? ???? ????????? average ??? ?????? ASCII ??? ?????????? ????????? ??? USERNAME.
            average += username[i];
    //???????????? ?? average ??? ??? 2.
    return average / 2;

}

//????????? ?????????? HashList.
void BuildHashList(HashListType *HList)
{
    //?????? ??????????.
    ListElm AnItem;
    FILE *input;
```

```
262      int nscan;
263
264      //????????? HashList.
265      CreateHashList(&(*HList));
266
267      //??????? ??????? ??? ????????.
268      input = fopen("I5F6.txt","r");
269
270      //?? ??? ???????,???????? ???????? ?????????.
271      if(input == NULL)
272          printf("Can't open file.\n");
273      //?? ??????? ????????
274      else
275          while(TRUE)
276          {
277              //?????????? ?? ???????? USERNAME ??? PASSWORD.
278              nscan = fscanf(input, "%s %s",AnItem.RecKey,AnItem.Password);
279
280              //???? ????????? ? ????????? ??? ??????? ,????? ??????????.
281              if(nscan == EOF) break;
282              //?? ??????? ???????? ???? ????????, ????????? ??????????? ????????? ??? ?????? ??? ???
???????????? ??????????.
283              if(nscan != 2 )
284              {
285                  printf("Improper file format.\n");
286                  break;
287              }
288              //?? ????? ??? ????
289              else
290                  //????????? ??? ????????? ??? ??????????? ??? HashList.
291                  AddRec(&(*HList),AnItem);
292          }
293  }
```