Tutorial 5 (Week 6)
Structures

Q1 (computeCircle)

A structure called circle is defined below. The structure consists of the radius of the circle and the (x,y) coordinates of its centre. A structure called circle is defined below. The structure consists of the radius of the circle and the (x,y) coordinates of its centre.

```
struct circle {
    double radius;
    double x;
    double y;
};
```

(a) Implement the function **intersect()** that returns 1 if two circles intersect, and 0 otherwise. Two circles intersect when the distance between their centres is less than or equal to the sum of their radii. The function prototype is given below:

```
int intersect(struct circle c1, struct circle c2);
```

(b) Implement the function **contain()** that returns 1 if *c1* contains *c2*, i.e. circle *c2* is found inside circle *c1*. Otherwise, the function returns 0. Circle *c1* contains circle *c2* when the radius of *c1* is larger than or equal to the sum of the radius of *c2* and the distance between the centres of *c1* and *c2*. The function prototype of contain() is given below:

```
int contain(struct circle *c1, struct circle *c2)
```

Write a program to test the functions.

Sample input and output sessions:

(1) Test Case 1

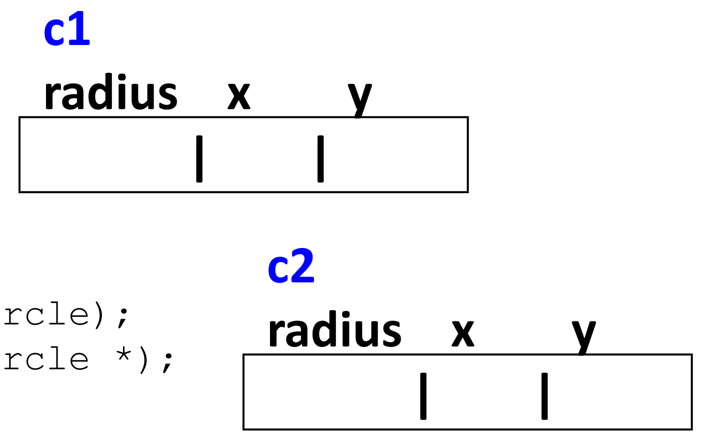
```
Enter circle 1 (radius x y):
10 5 5
Enter circle 2 (radius x y):
5 1 1
Circle intersection: 1
Circle contain: 0
```

(2) Test Case 2

```
Enter circle 1 (radius x y):
10 5 5
Enter circle 2 (radius x y):
1 1 1
Circle intersection: 1
Circle contain: 1
```

Q1 (computeCircle)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define INIT_VALUE -1000
struct circle {
   double radius;
   double x;
   double y;
};
int intersect(struct circle, struct circle);
int contain(struct circle *, struct circle *);
int main()
{
   struct circle c1, c2;
   int choice, result = INIT_VALUE;

   printf("Select one of the following options: \n");
   printf("1: intersect()\n");
   printf("2: contain()\n");
   printf("3: exit()\n");
   do {
      result=-1;
      printf("Enter your choice: \n");
      scanf("%d", &choice);
      switch (choice) {
```

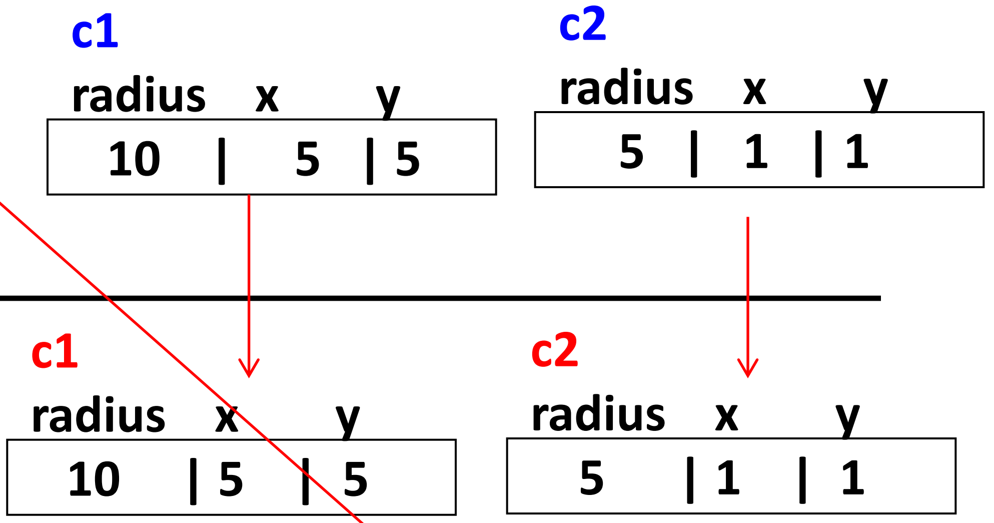


```
        case 1:
            printf("Enter circle 1 (radius x y): \n");
            scanf("%lf %lf %lf", &c1.radius, &c1.x, &c1.y);
            printf("Enter circle 2 (radius x y): \n");
            scanf("%lf %lf %lf", &c2.radius, &c2.x, &c2.y);
            result = intersect(c1, c2);
            if (result == 1)
               printf("intersect(): intersect\n");
            else if (result == 0)
               printf("intersect(): not intersect\n");
            else
               printf("intersect(): error\n");
            break;
         case 2:
            printf("Enter circle 1 (radius x y): \n");
            scanf("%lf %lf %lf", &c1.radius, &c1.x, &c1.y);
            printf("Enter circle 2 (radius x y): \n");
            scanf("%lf %lf %lf", &c2.radius, &c2.x, &c2.y);
            result = contain(&c1, &c2);
            if (result == 1)
               printf("contain(): contain\n");
            else if (result == 0)
               printf("contain(): not contain\n");
            else
               printf("contain(): error\n");
            break;
      }
   } while (choice < 3);
   return 0;
}
```

Use dot notation when accessing members of the structure.

```
int main()
{
   result = intersect(c1, c2);

}
```

Call by value

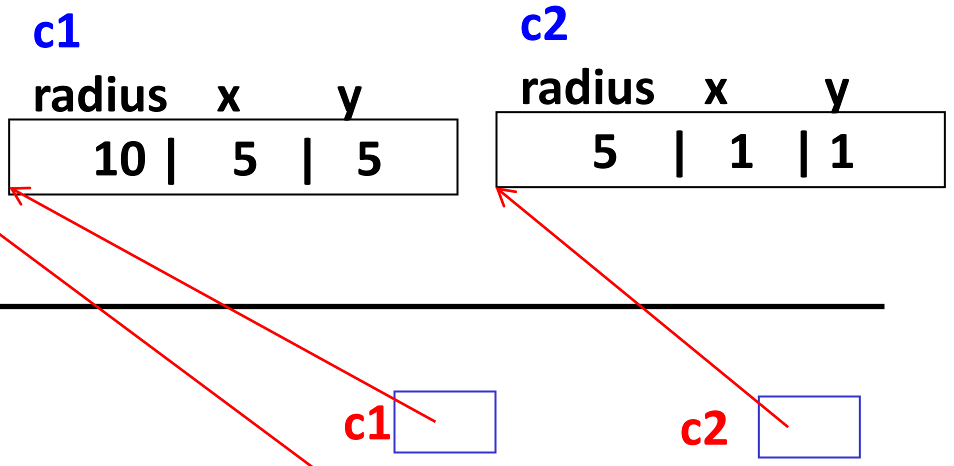


```
int intersect(struct circle c1, struct circle c2)
{
    double a, b;

    a = c1.x - c2.x;
    b = c1.y - c2.y;
    return (sqrt(a*a + b*b) <= (c1.radius + c2.radius));
}
```

Use dot notation when accessing members of the structure in this function.

```
int main()
{
  result = contain(&c1, &c2);

}
```

Call by reference



```
int contain(struct circle *c1, struct circle *c2)
{
  double a, b;

  a = c1->x - c2->x;
  b = c1->y - c2->y;
  return (c1->radius >=(c2->radius+sqrt(a*a+b*b)));
}
```

Use -> notation when accessing members of the structure in this function.

Q2 (computeAverage)

Assume the following structure is defined to represent a grade record of a student:

```
struct student{
      char name[20];              /* student name */
      double testScore;          /* test score */
      double examScore;         /* exam score */
      double total; /* total = (testScore+examScore)/2 */
 };
```

Write a C program to create a database of maximum 50 students using an array of structures.

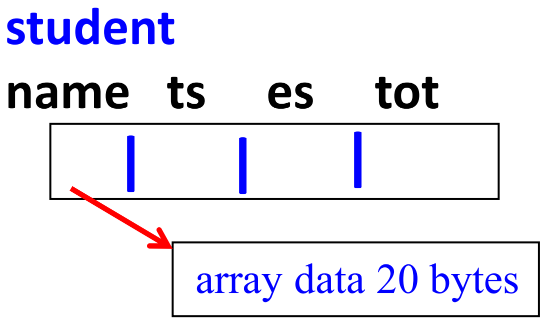
1. The program takes in the number of students in the class.
2. For each student, it takes in the test score and exam score. Then it computes and prints the **total score** for each student. The input will end when the student name is "**END**".
3. Then, it computes and prints the total score for each student, and computes and prints the **average score** of all students.

Sample input and output session:

```
Enter student name:
Hui Cheung
Enter test score:
34
Enter exam score:
46
Student Hui Cheung total = 40.00
Enter student name:
Tan May
Enter test score:
60
Enter exam score:
80
Student Tan May total = 70.00
Enter student name:
END
average(): 55.00
```

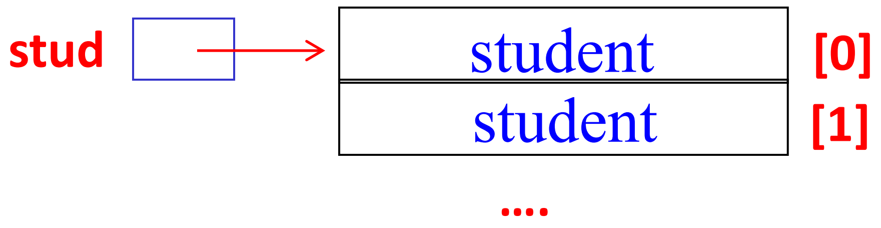
Q3 (computeAverage)

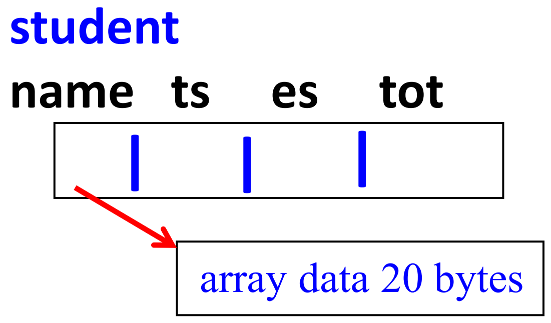
```
#include <stdio.h>
#include <string.h>
struct student{
   char name[20]; /* student name */
   double testScore; /* test score */
   double examScore; /* exam score */
   double total;  /* total = (testscore+examscore)/2 */
};
double average();
int main()
{
   printf("average(): %.2f\n", average());
   return 0;
}
```



Q3 (computeAverage)

```
double average(){
   struct student stud[50];
   double sum = 0;
   char *p; int i;
   i=0;
   printf("Enter student name: \n");
   fgets(stud[i].name, 50, stdin);
   if (p=strchr(t,'\n'))    *p = '\0';
   while (strcmp(stud[i].name, "END")!=0)
   {
      /* get scores */
      printf("Enter test score: \n");
      scanf("%lf", &stud[i].testScore);
      printf("Enter exam score: \n");
      scanf("%lf", &stud[i].examScore);
      /* compute total */
      stud[i].total = (stud[i].testScore + stud[i].examScore)/2;
      printf("Student %s total = %.2f\n", stud[i].name, stud[i].total);
      sum += stud[i].total;
      i++;
      printf("Enter student name: \n");
      scanf("\n");        // remove the remaining char in the buffer
      fgets(stud[i].name, 50, stdin);
      if (p=strchr(t,'\n'))    *p = '\0';
   }
   if (i != 0)
      return (sum/i);
   else
      return 0;
}
```





Q3 (phonebook)

Write a C program that implements the following three functions:

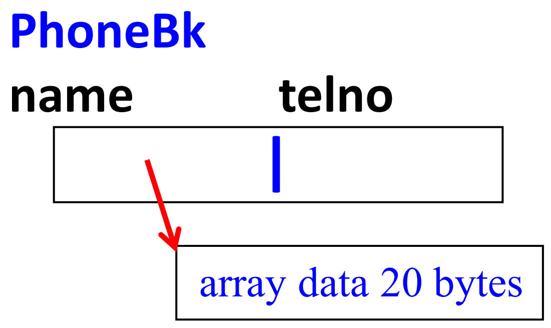
- The function **readin()**
  - ➢ It reads a number of persons' names and their corresponding telephone numbers, passes the data to the caller via the parameter **pb**, and returns the number of names that have entered.
  - ➢ The character '#' is used to indicate the **end of user input**.
- The function **printPB()**
  - ➢ It prints the phonebook information on the display.
  - ➢ It will print the message "Empty phonebook" if the phonebook list is empty.
- The function **search()**
  - ➢ It finds the telephone number of an input name ***target***, and then prints the name and telephone number on the screen.
  - ➢ If the input name cannot be found, then it will print an appropriate error message.

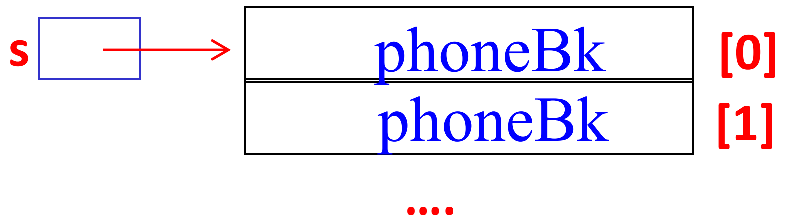
The prototypes of the three functions are given below:

```
void printPB(PhoneBk *pb, int size);
int readin(PhoneBk *pb);
void search(PhoneBk *pb, int size, char *target);
```

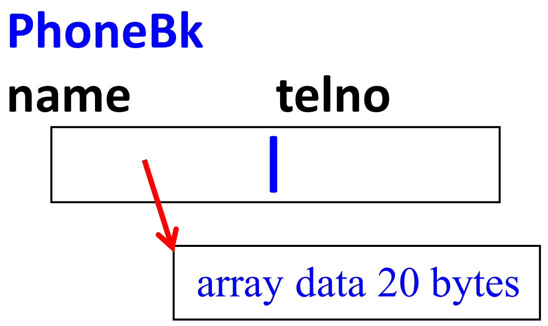
```
Enter your choice:
1
Enter name:
Hui Siu Cheung
Enter tel:
12345
Enter name:
Philip Fu
Enter tel:
23456
Enter name:
Chen Jing
Enter tel:
34567
Enter name:
#
Enter your choice:
3
The phonebook list:
Name: Hui Siu Cheung
Telno: 12345
Name: Philip Fu
Telno: 23456
Name: Chen Jing
Telno: 34567

Enter your choice:
2
Enter search name:
Philip Fu
Name = Philip Fu, Tel = 23456
```

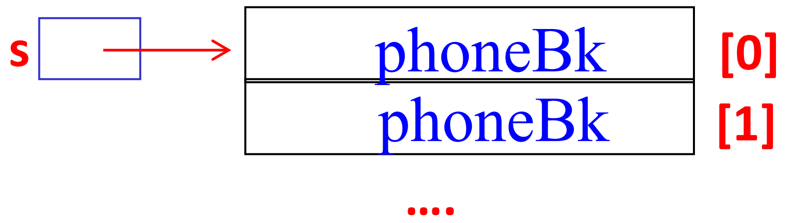
```
#include <stdio.h>
#include <string.h>
#define MAX 100
typedef struct {
   char name[20];
   int telno;
} PhoneBk;
void printPB(PhoneBk *pb, int size);
int readin(PhoneBk *pb);
void search(PhoneBk *pb, int size, char *target);
int main()
{
   PhoneBk s[MAX];
   char t[20], *p;
   int size=0, choice;

   char dummychar;


   printf("Select one of the following options: \n");
   printf("1: readin()\n");
   printf("2: search()\n");
   printf("3: printPB()\n");
   printf("4: exit()\n");
   do {
      printf("Enter your choice: \n");
      scanf("%d", &choice);
```



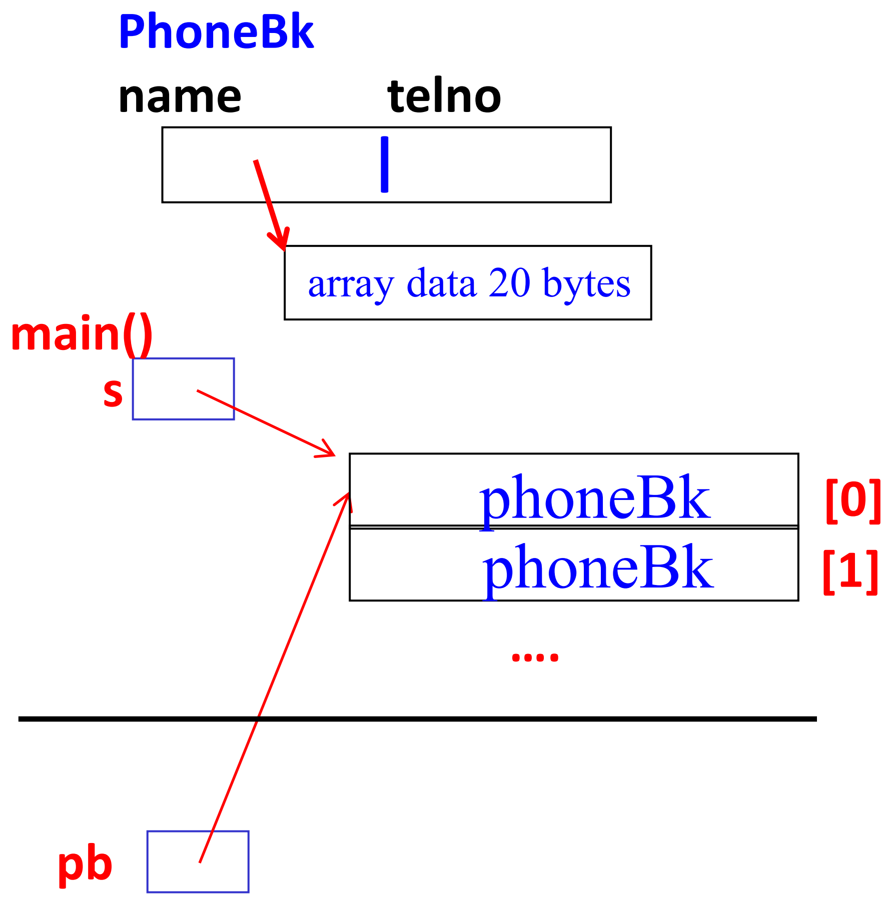


```
switch (choice) {
        case 1:
          scanf("%c", &dummychar);
          size = readin(s);
          break;
        case 2:
          scanf("%c", &dummychar);
          printf("Enter search name: \n");
          fgets(t, 20, stdin);
          if (p=strchr(t,'\n')) *p = '\0';
          search(s,size,t);
          break;
        case 3:
          printPB(s, size);
          break;
      }
   } while (choice < 4);
   return 0;
}
```

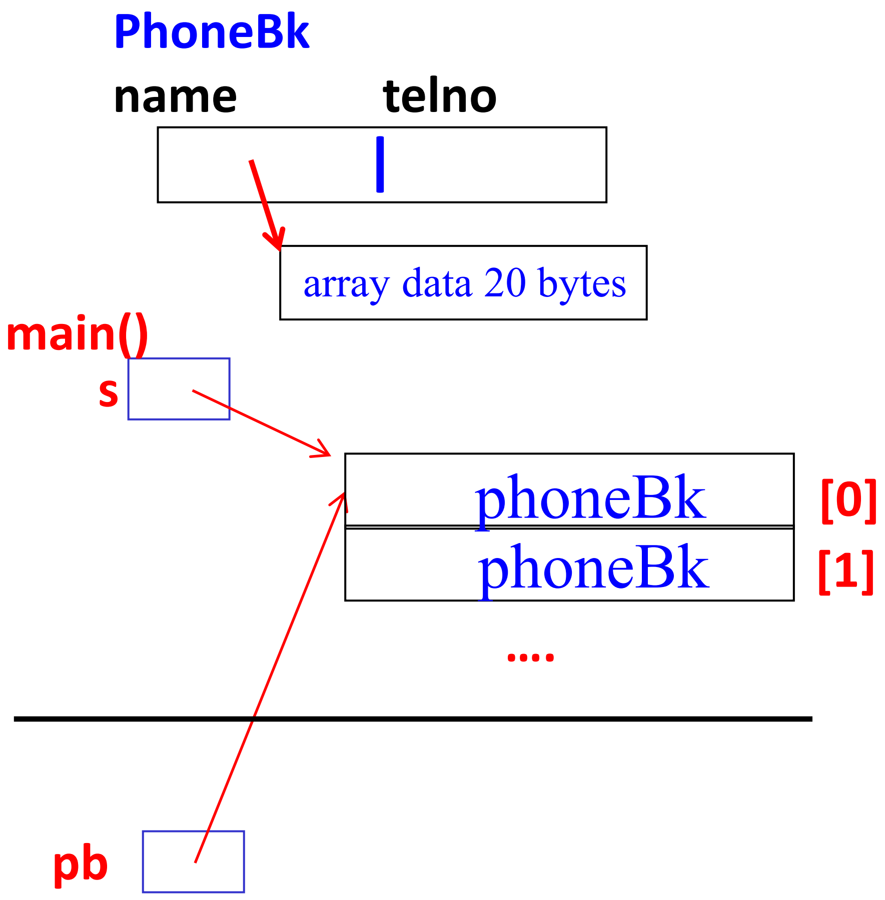




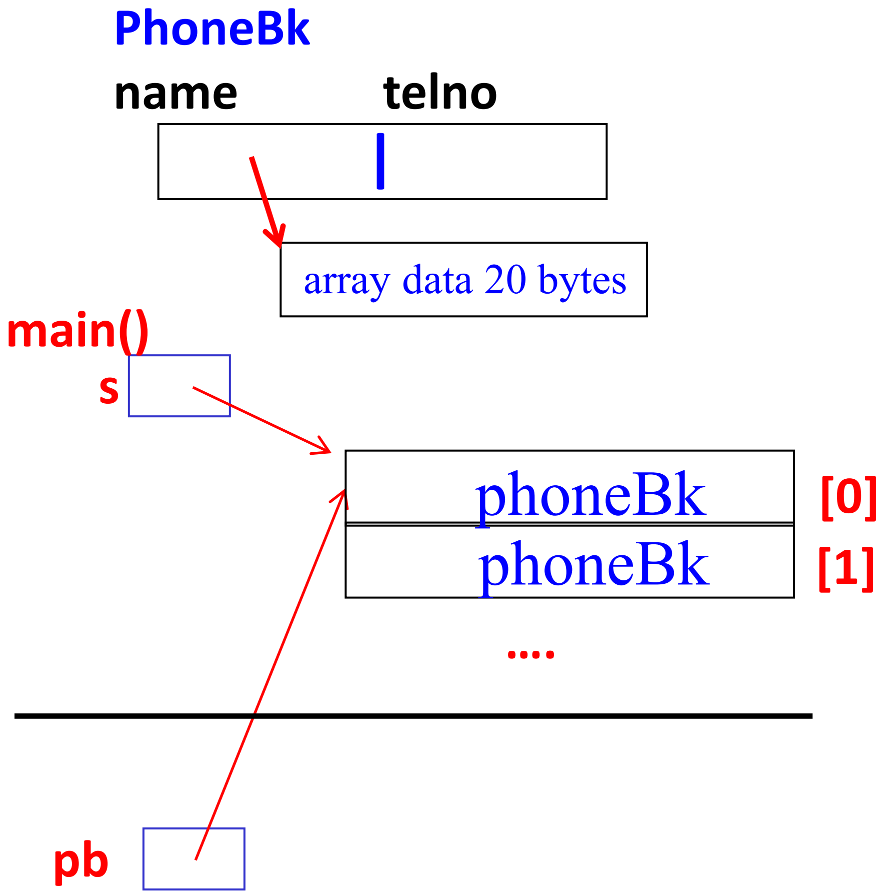
```
void printPB(PhoneBk *pb, int size)
{
  int i;

  printf("The phonebook list: \n");
  if (size==0)
    printf("Empty phonebook\n");
  else {
    for (i=0; i<size; i++) {
      printf("Name: %s\n", (pb+i)->name);
      printf("Telno: %d\n", (pb+i)->telno);
    }
  }
}
```



```
int readin(PhoneBk *pb)
{
  int size=0;
  char *p, dummy[80];

  while (1) {
    printf("Enter name: \n");
    fgets(pb->name, 20, stdin);
    if (p=strchr(pb->name,'\n')) *p = '\0';
    if (strcmp(pb->name,"#")==0)
      break;
    printf("Enter tel: \n");
    scanf("%d",&(pb->telno));
    fgets(dummy, 80, stdin);
    pb++;
    size++;
  }
  return size;
}
```



```
void search(PhoneBk *pb, int size, char *target)
{
  int i;

  for (i=0;i<size;i++,pb++) {
    if (strcmp(pb->name,target)==0){
      printf("Name = %s, Tel = %d\n",
             target,pb->telno);
      break;
    }
  }
  if (i==size)
    printf("Name not found!\n");
}
```


PhoneBk
name
telno
array data 20 bytes
main()
s
phoneBk
[0]
phoneBk
[1]
....
pb