Tutorial 5 - Structures

1. (**compouteCircle**) 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 is given below:

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

A sample program template is given below to test the functions:

```
#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;
    }
    int intersect(struct circle c1, struct circle c2)
    {
        /* Write your code here */
    }
    int contain(struct circle *c1, struct circle *c2)
    {
        /* Write your code here */
    }
```

Some sample input and output sessions are given below:

(1) Test Case 1:

```
Select one of the following options:
1: intersect()()
2: contain()
```

3: exit()
Enter your choice:
1
Enter circle 1 (radius x y):
10 5 5
Enter circle 2 (radius x y):
5 1 1
intersect(): intersect
Enter your choice:
3

(2) Test Case 2:
Select one of the following options:
1: intersect()()
2: contain()
3: exit()
Enter your choice:
2
Enter circle 1 (radius x y):
10 5 5
Enter circle 2 (radius x y):
1 1 1
contain(): contain
Enter your choice:
3

(3) Test Case 3:
Select one of the following options:
1: intersect()()
2: contain()
3: exit()
Enter your choice:
1
Enter circle 1 (radius x y):
1 5 5
Enter circle 2 (radius x y):
1 10 10
intersect(): not intersect
Enter your choice:
3

(4) Test Case 4:
Select one of the following options:
1: intersect()()
2: contain()
3: exit()
Enter your choice:
2

```
Enter circle 1 (radius x y):
1 5 5
Enter circle 2 (radius x y):
1 10 10
contain(): not contain
Enter your choice:
3
```

2. (**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 function average() that creates a database of maximum 50 students using an array of structures. The function reads in student name. For each student, it takes in the test score and exam score. Then it computes and prints the total score of the student. The input will end when the student name is "END". Then, it computes and returns the average score of all students to the calling function (i.e. main()). The calling function then prints the average score on the display. The function prototype is given below:

```
double average();
```

A sample program template is given below to test the function:

```
#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;
}
double average()
{
  /* Write your code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
```
Enter student name:
Hui S
Enter test score:
35.5
Enter exam score:
43.5
Student Hui S total = 39.50
Enter student name:
END
average(): 39.50
```

(2) Test Case 2:
```
Enter student name:
Hui S
Enter test score:
34
Enter exam score:
45
Student Hui S total = 39.50
Enter student name:
Fong A
Enter test score:
67
Enter exam score:
56
Student Fong A total = 61.50
Enter student name:
END
average(): 50.50
```

(3) Test Case 3:
```
Enter student name:
END
average(): 0.00
```

3. (**phoneBook**) Write a C program that implements the phoebook management system with the following three functions:

- The function readin() reads a number of persons' names and their corresponding telephone numbers, passes the data to the caller via the parameter p, and returns the number of names that have entered. The character '#' is used to indicate the end of user input.
- The function printPB() prints the phonebook information on the display. It will print the message "Empty phonebook" if the phonebook list is empty.

- The function search() 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 two functions are given below:

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);
```

The structure definition for PhoneBk is given below:

```
typedef struct {
  char name[20];     // a string
  int telno;         // an integer with 5 digits
} PhoneBk;
```

A sample program template is given below to test the functions:

```
#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)
    {
      /* Write your code here */
    }
    int readin(PhoneBk *pb)
    {
      /* Write your code here */
    }
    void search(PhoneBk *pb, int size, char *target)
    {
      /* Write your code here */
    }
```

Some test input and output sessions are given below:

(1) Test Case 1:

```
Select one of the following options:
1: readin()
2: search()
3: printPB()
4: exit()
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:
4

(2) Test Case 2:
<continue from Test Case 1>
Enter your choice:
2
Enter search name:
Philip Fu
Name = Philip Fu, Tel = 23456
Enter your choice:
4

(3) Test Case 3:
<continue from Test Case 1>
Enter your choice:
2
Enter search name:
Tommy Fu
Name not found!
Enter your choice:
4

(4) Test Case 4:
Select one of the following options:
1: readin()
2: search()
3: printPB()
4: exit()
Enter your choice:
1
Enter name:
#

Enter your choice:
3
The phonebook list:
Empty phonebook
Enter your choice:
4