

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) The function prototype of **contain()** is given below:

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

The function contain() 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. Implement the function contain().

Write a C program to test the functions.

Structures – Q1

Enter circle 1 (radius x y):

10 5 5

Enter circle 2 (radius x y):

5 1 1

Circle intersection: 1

Circle contain: 0

Continue ('y' or 'n') :

y

Enter circle 1 (radius x y):

10 5 5

Enter circle 2 (radius x y):

1 1 1

Circle intersection: 1

Circle contain: 1

Continue ('y' or 'n') :

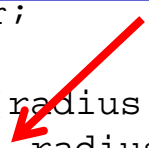
n

Structures – Q1

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
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;
    char repeat = 'y', dummychar;
    do {
        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);
        printf(" intersect(): %d\n", intersect(c1,
c2));
        printf("contain(): %d\n", contain(&c1, &c2));
        scanf("%c", &dummychar);
        printf("\nContinue ('y' or 'n') : \n");
        scanf("%c", &repeat);
    } while (repeat == 'y');

    return 0;
}
```

Use dot notation when accessing members of the structure.



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

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

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)));
}
```

A structure is defined to represent an arithmetic expression:

```
typedef struct {  
    float operand1, operand2;  
    char op;    /* operator '+', '-', '*', or '/' */  
} bexpression;
```

(a) Write a C function that computes the value of an expression and returns the result. For example, the function will return the value of 4/2 if in the structure passed to it, operand1 is 4, operator is '/' and operand2 is 2. The function prototype is given as:

float compute1(bexpression *expr*);

(b) Write another C function that performs the same computation with the following function prototype:

float compute2(bexpression **expr*);

Write a C program to test the functions.

Structures – Q2

Enter expression (op1 op2 op) :

4 8 +

compute1 = 12.000000

compute2 = 12.000000

Continue ('y' or 'n') :

y

Enter expression (op1 op2 op) :

8 4 /

compute1 = 2.000000

compute2 = 2.000000

Continue ('y' or 'n') : **y**

Enter expression (op1 op2 op) :

4 8 *

compute1 = 32.000000

compute2 = 32.000000

Continue ('y' or 'n') :

n

Structures – Q2

```
#include <stdio.h>
typedef struct {
    float operand1, operand2;
    char op;
} bexpression;


float compute1(bexpression expr);
float compute2(bexpression *expr);
int main()
{
    bexpression e;
    char repeat = 'y', dummychar;

    do {
        printf("Enter expression (op1 op2 op): \n");
        scanf("%f %f %c", &e.operand1, &e.operand2, &e.op);
        printf("compute1(): %.2f\n", compute1(e));
        printf("compute2(): %.2f\n", compute2(&e));
        scanf("%c", &dummychar);
        printf("\nContinue ('y' or 'n'): \n");
        scanf("%c", &repeat);
    } while (repeat == 'y');


    return 0;
}
```

Structures – Q2

```
float compute1(bexpression expr){  
    float result;  
    switch (expr.op) {  
        case '+': result = expr.operand1 + expr.operand2;  
            break;  
        case '-': result = expr.operand1 - expr.operand2;  
            break;  
        case '*': result = expr.operand1 * expr.operand2;  
            break;  
        case '/': result = expr.operand1 / expr.operand2;  
            break;  
    }  
    return result;  
}
```



```
float compute2(bexpression *expr)  
{  
    float result;  
    switch (expr->op) {  
        case '+': result = expr->operand1 + expr->operand2;  
            break;  
        case '-': result = expr->operand1 - expr->operand2;  
            break;  
        case '*': result = expr->operand1 * expr->operand2;  
            break;  
        case '/': result = expr->operand1 / expr->operand2;  
            break;  
    }  
    return result;  
}
```



Structures – Q3

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. The program takes in the number of students in the class.

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”.

Then, it computes and prints the total score for each student, and computes and prints the average score of all students.

Enter student name:

Hui Cheung

Enter test score:

34

Enter exam score:

46

Student Hui Cheung total = 40.000000

Enter student name:

Tan May

Enter test score:

60

Enter exam score:

80

Student Tan May total = 70.000000

Enter student name:

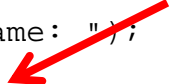
END

Overall average: 55.000000

Structures – Q3

```
#include <stdio.h>
struct student{
    char name[20]; /* student name */
    double testScore; /* test score */
    double examScore; /* exam score */
    double total; /* total score = (testScore+examScore)/2 */
};
double average();
int main(){
    printf("average(): %.2f\n", average());
    return 0;
}
double average(){
    struct student stud[50];
    double sum = 0;
    int i;
    char dummychar;
    /* get student scores */
    i=0;
    printf("Enter student name: ");
    gets(stud[i].name);
    while (strcmp(stud[i].name, "END")!=0){
        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("%c", &dummychar);
        gets(stud[i].name);
    }
    if (i != 0) return (sum/i);
    else return 0;
}
```

Use dot notation when
accessing members of
the structure.



Given the following information, write the code for the functions `getInput()`, `mayTakeLeave()` and `printList()`.

Structures – Q4

```
typedef struct {  
    int id;          /* staff identifier */  
    int totalLeave;   /* the total number of days of leave allowed */  
    int leaveTaken;  /* the number of days of leave taken so far */  
} leaveRecord;
```

(a) void getInput(leaveRecord list[], int *n);

Each line of the input has three integers representing one staff identifier, his/her total number of days of leave allowed and his/her number of days of leave taken so far respectively. The function will read the data into the array *list* until end of input and returns the number of records read through *n*. The function prototype is given as follows:

(b) int mayTakeLeave(leaveRecord list[], int id, int leave, int n);

It returns 1 if a leave application for *leave* days is approved. Staff member with identifier *id* is applying for *leave* days of leave. *n* is the number of staff in *list*. Approval will be given if the leave taken so far plus the number of days applied for is less than or equal to his total number of leave days allowed. If approval is not given, it returns 0. It will return -1 if no one in *list* has identifier *id*.

(c) void printList(leaveRecord list[], int n);

It prints the list of leave records of each staff. *n* is the number of staff in *list*.

Write a C program to test the three functions.

Enter the number of staff records:

2

Enter id, totalleave, leavetaken:

11 28 25

Enter id, totalleave, leavetaken:

12 28 6

The staff list:

id = 11, totalleave = 28, leave taken = 25

id = 12, totalleave = 28, leave taken = 6

Please input id, leave to be taken:

11 6

The staff 11 cannot take leave

Enter the number of staff records:

2

Enter id, totalleave, leavetaken:

11 28 25

Enter id, totalleave, leavetaken:

12 28 6

The staff list:

id = 11, totalleave = 28, leave taken = 25

id = 12, totalleave = 28, leave taken = 6

Please input id, leave to be taken:

12 6

8

The staff 12 can take leave

Structures – Q4

```
#include <stdio.h>
typedef struct {
    int id;                /* staff identifier */
    int totalLeave;         /* the total number of days of leave allowed */
    int leaveTaken;        /* the number of days of leave taken so far */
} leaveRecord;
int mayTakeLeave(leaveRecord list[], int id, int leave, int n);
void getInput(leaveRecord list[], int *n);
void printList(leaveRecord list[], int n);

int main()
{
    leaveRecord listRec[10];
    int len;
    int id, leave, canTake=-1;

    getInput(listRec, &len);
    printList(listRec, len);
    printf("Please input id, leave to be taken: \n");
    scanf("%d %d", &id, &leave);
    canTake = mayTakeLeave(listRec, id, leave, len);
    if (canTake != 0)
        printf("The staff %d can take leave\n", id);
    else
        printf("The staff %d cannot take leave\n", id);
    return 0;
}
```

Structures – Q4

```
void getInput( leaveRecord list[], int *n)
{
    int total;

    *n = 0;
    printf("Enter the number of staff records: \n");
    scanf("%d", &total);

    while ( (*n) != total) {
        printf("Enter id, totalleave, leavetaken: \n");
        scanf("%d %d %d", &list[*n].id, &list[*n].totalLeave,&list[*n].leaveTaken);
        (*n)++;
    }
}

int mayTakeLeave( leaveRecord list[], int id, int leave, int n)
{
    int p;

    for (p = 0; p < n; p++)
        if (list[p].id == id)
            return (list[p].totalLeave >= (list[p].leaveTaken + leave));
    return -1;
}

void printList( leaveRecord list[], int n)
{
    int p;

    printf("The staff list:\n");
    for (p = 0; p < n; p++)
        printf ("id = %d, totalleave = %d, leave taken = %d\n",
            list[p].id, list[p].totalLeave, list[p].leaveTaken);
}
```