

WEEK 6 - STRUCTURES

You are required to do the following:

1. **Lab Questions** – Please do the lab questions during the lab session. When doing your lab questions, please follow exactly the question requirements on program input/output as our automated assessment system is based on test cases using exact string matching on program input/output.
2. **Lab Assignment Questions** – Please do the assignment questions and submit your code to the online Automated Programming Assessment System (APAS) for grading.

Lab Tutor: For this lab-tutorial session, please discuss the solution for each question in the lab. You may allocate about 30 minutes for each question. No need to discuss the assignment questions.

Lab Questions

1. **(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.

```
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 program code here */
}

int contain(struct circle *c1, struct circle *c2)
{
    /* Write your program 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

```
- (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

```
- (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

```
- (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

```

2. (**computeExp**) 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);
```

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

```
#include <stdio.h>
typedef struct {
    float operand1, operand2;
    char op;
} bexpression;
float compute1(bexpression expr);
float compute2(bexpression *expr);
int main()
{
    bexpression e;
    int choice;

    printf("Select one of the following options: \n");
    printf("1: compute1()\n");
    printf("2: compute2()\n");
    printf("3: exit()\n");
    do {
        printf("Enter your choice: \n");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter expression (op1 op2 op): \n");
                scanf("%f %f %c", &e.operand1, &e.operand2, &e.op);
                printf("compute1(): %.2f\n", compute1(e));
                break;
            case 2:
                printf("Enter expression (op1 op2 op): \n");
                scanf("%f %f %c", &e.operand1, &e.operand2, &e.op);
                printf("compute2(): %.2f\n", compute2(&e));
                break;
        }
    } while (choice < 3);
    return 0;
}
float compute1(bexpression expr)
{
    /* Write your program code here */
}
float compute2(bexpression *expr)
{
    /* Write your program code here */
}
```

Some sample input and output sessions are given below:

- (1) Test Case 1:
 Select one of the following options:
 1: compute1()
 2: compute2()
 3: exit()
 Enter your choice:

- ```

1
Enter expression (op1 op2 op) :
5 8 +
compute1(): 13.00

```
- (2) Test Case 2:  
 Select one of the following options:  
 1: compute1()  
 2: compute2()  
 3: exit()  
 Enter your choice:  
2  
 Enter expression (op1 op2 op) :  
8 5 /  
 compute2(): 1.60
- (3) Test Case 3:  
 Select one of the following options:  
 1: compute1()  
 2: compute2()  
 3: exit()  
 Enter your choice:  
1  
 Enter expression (op1 op2 op) :  
5 8 \*  
 compute1(): 40.00
- (4) Test Case 4:  
 Select one of the following options:  
 1: compute1()  
 2: compute2()  
 3: exit()  
 Enter your choice:  
2  
 Enter expression (op1 op2 op) :  
8 5 -  
 compute2(): 3.00

3. **(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 program 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

```

4. (**mayTakeLeave**) Given the following structure definition, write the code for the functions `getInput()`, `mayTakeLeave()` and `printList()` with the following function prototypes:

```
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*.

(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*.

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

```
#include <stdio.h>
#define INIT_VALUE 1000
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=INIT_VALUE;
 int choice;

 printf("Select one of the following options: \n");
 printf("1: getInput()\n");
 printf("2: printList()\n");
 printf("3: mayTakeLeave()\n");
 printf("4: exit()\n");
 do {
 printf("Enter your choice: \n");
 scanf("%d", &choice);
 switch (choice) {
 case 1:
 getInput(listRec, &len);
 printList(listRec, len);
 break;
 case 2:
 mayTakeLeave(listRec, id, leave, len);
 break;
 case 3:
 mayTakeLeave(listRec, id, leave, len);
 break;
 case 4:
 exit(0);
 break;
 default:
 printf("Invalid choice\n");
 break;
 }
 } while (choice != 4);
}
```

```

 printList(listRec, len);
 break;
 case 3:
 printf("Please input id, leave to be taken: \n");
 scanf("%d %d", &id, &leave);
 canTake = mayTakeLeave(listRec, id, leave, len);
 if (canTake == 1)
 printf("The staff %d can take leave\n", id);
 else if (canTake == 0)
 printf("The staff %d cannot take leave\n", id);
 else if (canTake == -1)
 printf("The staff %d is not in the list\n", id);
 else
 printf("Error!");
 break;
 }
} while (choice < 4);
return 0;
}
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);
}
void getInput(leaveRecord list[], int *n)
{
 /* Write your program code here */
}
int mayTakeLeave(leaveRecord list[], int id, int leave, int n)
{
 /* Write your program code here */
}

```

Some sample input and output sessions are given below:

(1) Test Case 1:  
 Select one of the following options:  
 1: getInput()  
 2: printList()  
 3: mayTakeLeave()  
 4: exit()  
 Enter your choice:  
1  
 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  
 Enter your choice:  
3  
 Please input id, leave to be taken:



11 6

The staff 11 cannot take leave  
Enter your choice:

4

(2) Test Case 2:

Select one of the following options:

1: getInput()  
2: printList()  
3: mayTakeLeave()  
4: exit()

Enter your choice:

1

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

Enter your choice:

3

Please input id, leave to be taken:

12 6

The staff 12 can take leave

Enter your choice:

4

(3) Test Case 3:

Select one of the following options:

1: getInput()  
2: printList()  
3: mayTakeLeave()  
4: exit()

Enter your choice:

1

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

Enter your choice:

3

Please input id, leave to be taken:

13 6

The staff 13 is not in the list

Enter your choice:

4

(4) Test Case 4:

Select one of the following options:

1: getInput()  
2: printList()

```
3: mayTakeLeave()
4: exit()
Enter your choice:
1
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
Enter your choice:
4
```