Tutorial 5 – Structures – Suggested Answers

1. 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);
```

Suggested Answer:

```
#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)
{
  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)));
}
```

2. 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);
```

Suggested Answer:

```
#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)
{
    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;
}
```

3. 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*.

Suggested Answer:

```
#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:
          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 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);
}
```