

Week 6 Lab Tutorial: Structures – Suggested Solutions

Lab Questions

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;
}
```

```

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));
}
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)));
}

```

Q2: (computeExp)

```

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

    printf("Select one of the following options: \n");
    printf("1: computel()\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("computel(): %.2f\n", computel(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 computel(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;
}

```

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;
}
double average()
{
    struct student stud[50];
    double sum = 0;
    int i;

    /* get student scores */
    i=0;
    printf("Enter student name: \n");
    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("\n");
    }
}

```

```

        gets(stud[i].name);
    }
    if (i != 0)
        return (sum/i);
    else
        return 0;
}

```

Q4: (mayTakeLeave)

```

#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 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)
{
    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;
}

```