Tutorial 5 - Structures

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);
```

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);
```

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*.