CS100 Tutorial 4 (Fall semester, 2018)

1. A structure called circle is defined below. The structure consists of the radius of the circle and values of x and y for the coordinates of the circle's center (x, y).

```
struct circle {
   double radius;
   double x;
   double y;
};
```

(1) Implement the intersect() function which returns 1 if two circles intersect, and 0 otherwise. Two circles intersect when the distance between their centers 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);
```

(2) 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 centers of c1 and c2. Implement contain(). The function prototype of contain() is given below:

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

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

```
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;


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 the end of input and returns
the number of records read through ‘n’ */
```

```
int mayTakeLeave(leaveRecord list[], int id, int leave, int n);
/* 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 members 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, return 0. If no one in ‘list’ has identifier ‘id’, return -1. */
```

3. A function sumUp() is defined as

   sumUp(1) = 1
   sumUp(n) = n + sumUp(n – 1),   if n > 1

   Here, we assume that n is a positive integer.
   Write a recursive function, sumUp(), where the function prototype is:

   ```
   int sumUp(int n);
   ```

   Write the iterative version of the function (i.e. using looping), sumUp2(), which returns the same value as sumUp(). Its prototype is given below:

   ```
   int sumUp2(int n);
   ```

4. The following functions compute x to the power of n, where n may be any integer value. Use the recursive approach to implement these functions. The function power1() returns the computed result while power2() passes the result back through the third parameter y. The function prototypes are given below. No error checking on x is needed.

   ```
   float power1(float x, int n);
   void power2(float x, int n, float *y);
   ```