```c
// J Hundley
// assign07
// March 19, 2015

/******USING USER CREATED FUNCTIONS*****
 call-by-value and call-by-reference functions
 Read the input values from a text data file
 Compute the distance
 While not end of file read one set of input values
 count the balloon
 find the velcocity
 compute the distance
count hit and add water to amount in the pool
 keep up with the last balloon number that hit the pool
 Display the number of hits, number thrown, and percent hits.
 Display the number of the balloon that caused the pool water to reach capacity
 Display a message saying how many gallons spilled over the edge of the pool
*/
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

// FUNCTION PROTOTYPES=================================================
double getBalloonVolume( int diameter );
double compDistance( double degrees, double velocity, double thrower_ht );
void printResults( int numHits, int numBalloons, int holdBalloonCount, double totalWater
);
void isHit( double *poolWater, double balloonVolume, int *numHits, int *holdBalloonCount,
int numBalloons );
void addOne( int *count );

//*****CONSTANT*****
#define BALCONY_HT  12.0     // balcony height in feet
#define G           32.0     // gravitational acceleration
#define PI          3.14159
#define POOL_RADIUS 1.0      // diameter of pool in feet
#define POOL_CENTER 35.0     // distance to the center of pool in feet
#define CAPACITY    7.0      // pool capacity in gallons

#define FILENAME    "balloonValues.txt"

int main()
{
// Problem Inputs:
   double theta,         // balloon launch angle (theta) in degrees
          velocity,      // balloon launch velocity (v) in ft/sec
          thrower_ht;    // thrower's height in feet
   int    diameter;      // diameter of balloon in inches
// Problem Outputs:
   int    numBalloons,   // number of balloons thrown
          numHits;       // number of balloons that hit the pool
// Other variables:
   double balloonVolume, // amount of water in a balloon in gallons
          poolWater,     // cumulative gallons of water in pool
          totalWater;    // cumulative gallons of water in all balloons
   double radians,       // angle in radians
          distance;      // distance a water balloon travels in feet
   int    holdBalloonCount; // hold the last balloon that hit the pool

   FILE * filePtr;       // file pointer
```

```
//*****INPUT*****
//open input data file
   filePtr = fopen(FILENAME,"r");
// check for good file open
   if (filePtr == NULL)
      printf("File Open Error");
   else // good file open continue program
   {
   //*****INITIALIZATION*****
      numBalloons = 0;  // number of balloons thrown
      numHits     = 0;  // number of balloons that hit the pool
      poolWater   = 0;  // cumulative gallons of water in pool
      totalWater  = 0;  // cumulative gallons of water in all balloons

   //*****READ A DATA FILE*****
   // get the balloon input values from the data file on at a time
      while(fscanf(filePtr, "%lf %lf %lf %d", &theta, &velocity, &thrower_ht, &diameter) != EOF)
      {
         addOne(&numBalloons);     // count balloon
         // use balloon diameter to determine volume
         balloonVolume = getBalloonVolume( diameter );

      //*****COMPUTATION*****
      // compute total water
         totalWater = totalWater + balloonVolume;
      // compute distance
         distance = compDistance( theta,  velocity, thrower_ht );
      // is it a hit?
         if (distance >= POOL_CENTER - POOL_RADIUS && distance <= POOL_CENTER + POOL_RADIUS)
            isHit( &poolWater, balloonVolume, &numHits, &holdBalloonCount, numBalloons );
      } // end while loop

      //*****OUTPUT*****
      printResults( numHits, numBalloons, holdBalloonCount, totalWater );
   } //end good open
   return 0;
}
// FUNCTION DEFINITIONS=====================================================
double getBalloonVolume( int diameter ){
   double volume;
       // use balloon diameter to determine volume
   switch (diameter)
   {
      case 3: volume = 0.1;
         break;
      case 4: volume = 0.2;
         break;
      case 5: volume = 0.3;
         break;
      case 6: volume = 0.55;
         break;
      case 7: volume = 0.8;
         break;
      case 8: volume = 1.25;
         break;
      case 9: volume = 1.7;
   } // no default needed because of input validation above
   return volume;
}
```

```c
double compDistance( double degrees, double velocity, double thrower_ht ){
   double part1,part2,part3, // partial result holders
          distance, radians;
   // convert degrees to radians
   radians = degrees * (PI/180);
   // compute distance
   part1    = velocity*cos(radians)/G;
   part2    = velocity*sin(radians);
   part3    = 2*G*(BALCONY_HT+thrower_ht);
   distance = part1 * (part2 + sqrt(part2*part2 + part3));
   return distance;
}

void printResults( int numHits, int numBalloons, int holdBalloonCount, double totalWater
){
   //*****OUTPUT*****
   printf("%d balloons hit the pool. \n", numHits);
   printf("%d balloons were thrown. \n", numBalloons);
   printf("%.2f%% balloons hit the pool. \n", (double)numHits/numBalloons*100);
   printf("balloon #%d filled the pool\n", holdBalloonCount);
   printf("%.2f gallons of water spilled over the edge pool.\n",totalWater-CAPACITY);
}

void isHit( double *poolWater, double balloonVolume, int *numHits, int *holdBalloonCount,
int numBalloons ){
   *poolWater =+ balloonVolume;         // add balloon volume
   addOne(numHits);                     // count hit
   if (*poolWater < CAPACITY)
      *holdBalloonCount = numBalloons;  // remember the balloon that hit the pool before
it was full
}

void addOne(int *count){
   *count = *count + 1;
   // *count ++;  ++ doesn't work with pointers
}
```

*Read all instructions before beginning your work.*

COMP1200-C - Assign 07
Due midnight – Thursday – March 12, 2015
**Submit `assign07.c` via Canvas**

*NOTE:*
*Your submitted file(s) MUST be spelled and cased as instructed.*
[-5 points for not doing so.]

***Before you start writing your program**:*

Save a copy of the `balloonValues.txt` data file from the Assign05 Announcement and in your COMP1200/assign07 folder. If you do not have folders set up for your assignment files, this is a good time to start. Your assign07.c will look in the folder where it is saved for the data file. A development plan is a process that guides you through solving a problem and creating an algorithm. Create your own algorithm and use it as comments throughout your program. Use section comments to group your statements as well as comments from your algorithm.

***Problem*:**

***Program*: assign07.c**

On a hot Saturday afternoon, you and your friends notice an empty baby swimming pool on the lawn of your apartment complex. So, why not see if you can fill it with water from water balloons thrown from your second floor balcony.

You will modify your assign06.c by adding two call-by-reference user-created functions. Include the following function prototypes in your assign07.c. You may modify the variable names but not the function names, return types, or parameter order, quantity, and data type.

```
// FUNCTION PROTOTYPES=======================================================
void isHit( double *poolWater, double balloonVolume, int *numHits,
            int *holdBalloonCount, int numBalloons );
```

Add the balloon water to the pool water; count hit; remember the balloon that filled the pool.

```
void addOne( int *count );
```

Add one to a count. Use for counting balloons and hits.

***Problem Constants*:**

See previous assignment.

***Problem Inputs*:**

See previous assignment.

***Problem Outputs*:**

See previous assignment.

***Other variables*:**

See previous assignment.

New commands:
call-by-reference functions
pointer type paramters

***Instructions:***

- ☐ See Standards for Documentation of C Programs on the Resources page on Canvas.
- ☐ Insert comments at the top and throughout each file.
  - o Include the follow comments at the beginning of this (and ALL) files.
    - // submitter's name, **GROUP #** — **Grade of ZERO for files with submitter name not part of Canvas group**
    - // other group members' names — **Type "none" if submitting alone.**
    - // assignment number — **Zero points for comments if no collaboration statement**
    - // date you completed the assignment
    - **// statement(s) about collaboration**
    - // a short narrative about what the file does
  - o Use the algorithm given as comments throughout your program.
- ☐ Use descriptive variable names.
- ☐ Use Sample Input/Output as a guide.
- ☐ Use **Generate CSD** to ensure correct indenting.
- ☐ Represent ALL given values as constants.
- ☐ See previous assignment.

*-5 points for absence of any of these required comments at the top at the top of each file.*

**If you do not submit individually,
there will be a 5 POINTS PENALTY for not joining a group.
Groups can be 2-4 students.
DO NOT join a group unless you have worked with the other members. If you do, you will be removed from the group and given the grade of zero.**

***Sample Input/Output:***

Same as previous assignment.

***Submit via* Canvas:**

`assign07.c` C program file

*NOTE: Your submitted file(s) MUST be spelled and cased as instructed.* [-5 points per file for not doing so.]