Computing Machinery I
Assignment 4

Structures and Subroutines

Create an ARMv8 assembly language program that implements the following program:

```
#define FALSE  0
#define TRUE   1

struct point {
  int x, y, z;
};

struct sphere {
  struct point origin;
  int radius;
};


struct sphere newSphere()
{
  struct sphere s;

  s.origin.x = s.origin.y = s.origin.z = 0;
  s.radius = 1;

  return s;
}

void move(struct sphere *s, int deltaX, int deltaY, int deltaZ)
{
  s->origin.x += deltaX;
  s->origin.y += deltaY;
  s->origin.z += deltaZ;
}

void expand(struct sphere *s, int factor)
{
  s->radius *= factor;
}

int equal(struct sphere *s1, struct sphere *s2)
{
  int result = FALSE;

  if (s1->origin.x == s2->origin.x) {
    if (s1->origin.y == s2->origin.y) {
      if (s1->origin.z == s2->origin.z) {
        if (s1->radius == s2->radius) {
          result = TRUE;
        }
      }
    }
  }

  return result;
}
```

```
void printSphere(char *name, struct sphere *s)
{
  printf("Sphere %s origin = (%d, %d, %d)  radius = %d\n",
         name, s->origin.x, s->origin.y, s->origin.z, s->radius);
}

main()
{
  struct sphere first, second;

  first = newSphere();
  second = newSphere();

  printf("\nInitial sphere values:\n");
  printSphere("first", &first);
  printSphere("second", &second);

  if (equal(&first, &second)) {
    move(&first, -5, 3, 2);
    expand(&second, 8);
  }

  printf("\nChanged sphere values:\n");
  printSphere("first", &first);
  printSphere("second", &second);
}
```

Implement all the subroutines above as unoptimized closed subroutines, using stack variables to store all local variables. Note that the function newSphere() must have a local variable (called *s*) which is returned by value to main(), where it is assigned to the local variables *first* and *second*. In other words, create code similar to what the C compiler produces, even if it seems inefficient. Name the program *assign4.asm*.

Also run the program in *gdb*, displaying the values of *first* and *second* after they have been set by function calls. You should show that the functions are working as expected. Capture the *gdb* session using the *script* UNIX command, and name the output file *script.txt*.

Other Requirements

Make sure your code is readable and fully documented, including identifying information at the top of each file. You must comment each line of assembly code. Your code should also be well designed: make sure it is well organized, clear, and concise.

New Skills Needed for this Assignment:

- Implementation of structs and nested structs
- Implementation of subroutines in assembly
- Returning structs by value from functions
- Use of pointers as arguments to subroutines

Submit the following:

1. Your assembly source code file for the program, and your script output file. Use the *Assignment 4* Dropbox Folder in D2L to submit electronically. The TA will assemble and run your program to test it. Be sure to name your program and script file as described above.

Computing Machinery I
Assignment 4 Grading

Student:_____

Functionality			
newSphere() function	6	_____	
move() function	4	_____	
expand() function	4	_____	
equal() function	6	_____	
printSphere() function	4	_____	
main() function	6	_____	
Correct implementation of structs	4	_____	
Correct use of stack variables	4	_____	
Script showing *gdb* session	2	_____	
Complete documentation and commenting	4	_____	
Design quality	2	_____	
Total	**46**	_____	_____%