Computing Machinery I
Assignment 4

Structures and Subroutines

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

```c
#define FALSE  0
#define TRUE   1

struct coord {
  int x, y;
};

struct size {
  int width, length;
};

struct pyramid {
  struct coord center;
  struct size base;
  int height;
  int volume;
};


struct pyramid newPyramid(int width, int length, int height)
{
  struct pyramid p;

  p.center.x = 0;
  p.center.y = 0;
  p.base.width = width;
  p.base.length = length;
  p.height = height;
  p.volume = (p.base.width * p.base.length * p.height) / 3;

  return p;
}

void relocate(struct pyramid *p, int deltaX, int deltaY)
{
  p->center.x += deltaX;
  p->center.y += deltaY;
}

void expand(struct pyramid *p, int factor)
{
  p->base.width *= factor;
  p->base.length *= factor;
  p->height *= factor;
  p->volume = (p->base.width * p->base.length * p->height) / 3;
}

void printPyramid(char *name, struct pyramid *p)
{
  printf("Pyramid %s\n", name);
  printf("\tCenter = (%d, %d)\n", p->center.x, p->center.y);
  printf("\tBase width = %d  Base length = %d\n", p->base.width, p->base.length);
  printf("\tHeight = %d\n", p->height);
  printf("\tVolume = %d\n\n", p->volume);
}
```

```
int equalSize(struct pyramid *p1, struct pyramid *p2)
{
  int result = FALSE;

  if (p1->base.width == p2->base.width) {
    if (p1->base.length == p2->base.length) {
      if (p1->height == p2->height) {
        result = TRUE;
      }
    }
  }

  return result;
}

int main()
{
  struct pyramid khafre, cheops;

  khafre = newPyramid(10, 10, 9);
  cheops = newPyramid(15, 15, 18);

  printf("Initial pyramid values:\n");
  printPyramid("khafre", &khafre);
  printPyramid("cheops", &cheops);

  if (!equalSize(&khafre, &cheops)) {
    expand(&cheops, 9);
    relocate(&cheops, 27, -10);
    relocate(&khafre, -23, 17);
  }

  printf("\nNew pyramid values:\n");
  printPyramid("khafre", &khafre);
  printPyramid("cheops", &cheops);
}
```

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

Also run the program in *gdb*, displaying the values of *khafre* and *cheops* 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*. Name your program *assign4.asm*.

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

newPyramid() function	8	_____	
relocate() function	4	_____	
expand() function	6	_____	
equalSize() function	6	_____	
printPyramid() function	6	_____	
main() function	9	_____	
Correct implementation of structs	4	_____	
Correct use of stack variables	4	_____	
Script showing *gdb* session	2	_____	
Complete documentation and commenting	4	_____	
Formatting (use of columns and white space)	4	_____	
Design quality	2	_____	
Total	**59**	_____	_____%