CS 221 C and Systems Programming
Assignment 5

Due at 11:59 pm on October 23, 2015

1 Pointers to Structures (30 marks)

The following structures are designed to store information about objects on a graphics screen:

```
struct point {int x, y; };
struct rectangle { struct point upper_left, lower_right; };
```

A point structure stores the `x` and `y` coordinates of a pointe on the screen. A `rectangle` structure stores the coordinates of the upper left and lower right corners of a `rectangle`. Write the following function to perform the specified tasks:

- `int area(const struct rectangle* r)`: this function returns the area of `r`.
- `struct point center(const struct rectangle* r)`: this function returns the center point of `r`. If either the `x` or `y` coordinate of the center is not an integer, store its truncated value in the point structure.
- `struct rectangle move(const struct rectangle* r, int addx, int addy)`: this function moves `r` by `addx` units in the `x` direction and `addy` units in the `y` direction, returning the modified version of `r`.
- `int inside(const struct rectangle* r, const struct point* p)`: this function returns 1 if `point p` lies within `rectangle r` and 0 otherwise.

Use the following `main` function to test your code.

```
int main()
{
  struct rectangle box = {{2,8},{10,3}};
  struct point p = {3,6};
  printf("area is %d\n", area(&box));
  struct point c = center(&box);
  printf("center is (%d,%d)\n", c.x, c.y);
  struct rectangle box2 = move(&box, 3, 6);
  printf("moved to (%d,%d) and (%d,%d)\n", box2.upper_left.x, box2.upper_left.y,
box2.lower_right.x, box2.lower_right.y);
  printf("inside function test result is %d\n", inside(&box,&p));
}
```

Name your file `struct.c`.

2 Unions (20 marks)

Given the following structure `point` and `shape` definitions:

```
struct point { int x, y; };
struct shape {
   enum {RECTANGLE, CIRCLE} shape_type;
   struct point center; //coordinates of center
   union {
      struct {
         int height, width;
      } rectangle;
      struct {
         int radius;
      } circle;
   } u;
};
```

Write the following function to perform the specified tasks:

- `void display(const struct shape s)`: this function displays the contents of `s` based on its `shape_type`. If it is a `RECTANGLE`, display `x, y` center point and the `height` and `width`. If it is a `CIRCLE`, display `x, y` center point and the `radius`.
- `double area(struct shape* s)`: this function returns the area of `s`.
- `struct shape move(const struct shape* s, int addx, int addy)`: this function moves `s` by `addx` units in the `x` direction and `addy` units in the `y` direction, returning the modified version of `s`.
- `struct shape scale(const struct shape* s, double c)`: this function returns the modified version of `s` which is scaled by the factor `c`.

Use the following `main` function to test your functions.

```
int main()
{
   struct shape a={RECTANGLE, {0,0}, {3,4}};
   struct shape b={CIRCLE, {0,0}, {5}};
   display(a);
   display(b);
   printf("a area is %f\n", area(&a));
   printf("b area is %f\n", area(&b));
   display(move(&a, -5, -2));
   display(move(&b, -5, -2));
   display(scale(&a,1.5));
   display(scale(&b,0.8));
}
```

Name your file `union.c`.

3 Self-referential Structures (50 marks)

Write a C program that reads a series of quotes from the users and sorts them according to the number of words of each quote. Here is a program running example:

```
   Enter a quote or Q to quit:  Imagination is more important than knowledge.
   Enter a quote or Q to quit:  It's not that I'm so smart, it's just that I stay with
problems longer.
   Enter a quote or Q to quit:  Only a life lived for others is a life worthwhile.
   Enter a quote or Q to quit:  Q

   In sorted order:
   Imagination is more important than knowledge.
   Only a life lived for others is a life worthwhile.
   It's not that I'm so smart, it's just that I stay with problems longer.
```

Your program should store the quotes in a linked list of `struct entry`, which contains a quote and the number of words of that quote.

```
struct entry {
   char *quote;
   int count;
   struct entry* next;
}
```

Each quote is no longer than 100 characters. The maximum number of quote (`MAXENTRY`) is 50. When no input (a new line) is entered at a prompt, ignore it and prompt for the next quote. Similarly, ignore the quote that is longer than 100 characters. Moreover, you need to validate that there is no duplicates in the linked list.

Implement your program as 4 functions: `main`, `read_quotes`, `process_quotes` and `print_result`. Use a header file to store the data shared by all files. Name your files as the following: `header.h`, `main.c`, `read.c`, `process.c` and `print.c`.

Submission Instructions

Similar to Assignment 1, you will generate 1 `typescript` file for compiling and running `struct.c`, `union.c`, `header.h`, `main.c`, `read.c`, `process.c` and `print.c`. Zip your files as follows: `zip ass5.zip struct.c union.c header.h main.c read.c process.c print.c typescript`. Finally, submit your `ass5.zip` to `Canvas`.