Name: ____

CS246 Sample Exam #2
March 28, 2017

This exam is open-book/open-note. You may use any printed resources you like. You may ***not*** use any computing devices, such as laptops, phones, or calculators.

This sample exam has ***not*** been length-tested. It may be shorter or longer than the real thing.

When writing code, the logic behind your code is more important than syntactic accuracy. In other words, pay attention to getting your loops right, not your semicolons.

1. For each expression below, write down the type of the expression, or write "invalid" if the expression is invalid. Assume the following declarations:

```
int i;
int* p;
int* q;
```

   a. `i` ________________

   b. `&i` ________________

   c. `p[0]` ________________

   d. `p + 2` ________________

   e. `&p` ________________

2. Consider the following stretch of code. Give the values of the expressions below after this code is run.

```
int i = 5;
int j = 10;
int* p = &i;
int* q = &i;

i++;
(*p)++;
j = *q;
(*q)++;
j--;
q = &j;
(*q)--;
```

a. `i` ____________

b. `j` ____________

c. `*p` ____________

d. `*q` ____________

3. Consider the following function and stretch of code. Give the values of the expressions below after the code is run.

```
int frob(int x, int* y)
{
     *y = x;
     x = 10;
     y = &x;
     return x + 5;
}

int a = 8;
int b = 6;

frob(a, &b);

int c = 17;

frob(c, &c);
```

a. `a` ____________________

b. `b` ____________________

c. `c` ____________________

4. Write a function that computes the first and last digit of a number, placing the results in the pointers provided.

```
// precondition: first and last point to valid ints
// postcondition: *first is the first (leftmost) digit in n;
//                *last is the last (rightmost) digit in n
void get_digits(int n, int* first, int* last)
{



}
```

5. Write a function that computes the sum of all the `int`s in a two-dimensional array. Other than in the function header, use no brackets. Additionally, use only one loop.

```
// precondition: arr is a m-by-n array
// postcondition: returns the sum of all the ints in the array
int sum(int m, int n, int arr[m][n])
{







}
```

6. Write a function to copy a string from one memory location to another. The function returns whether there is enough room to store the string in the destination memory.

```
// precondition: `from` points to a valid C string
//               `to` points to a region of memory `n` chars long
// postcondition: if there is enough space, `to` points to a copy
//                of the string in `from` and this function
//                returns true. Otherwise, this function returns
//                false.
bool string_copy(char* from, int n, char to[n])
{




}
```

7. Write a function to compute the geometric center of a collection of points. A point is represented by the `point` struct below. To compute a geometric center, simply average all the points. That is, the *x*-coordinate of the center is the average of the *x*-coordinates of all the points, and similar for the *y*-coordinate. You may find it useful to write a helper function.

```
typedef struct point {
     float x;
     float y;
} point;

// precondition: points is an array of n points
// postcondition: returns the geometric center of the points
point get_center(int n, point points[n])
{




}
```