

Assignment 3

ALECK ZHAO

September 26, 2016

Code Puzzles

1.

```
double x = 3.1415926535;
for (int i = 0; i < 8; i++) {
    int y = x;
    printf("%d", y);
    x -= y;
    x *= 10.0;
}
```

Answer. This prints out

31415926

2.

```
int i = 0;
int j = 1;
int k = -1;
bool a = true;
bool b = false;

if (b = true)
    printf("1");
if (j || (a = false))
    printf("2");

if (i || a)
    printf("3");
else
    printf("4");
    printf("5");

if (j == k) {
    printf("6");
} else if (b && k) {
    printf("7");
} else {
    printf("8");
}

for (i = 0; i < 5; i++)
{
    j++;
}

if (i > j) {
    printf("9");
}
```

Answer. This prints out

123579

```
3.   int y = 7;
      printf("%d\n", y);
      for (int x = 0; x < 10; x++) {
          int y = x * 2;
          printf("%d ", y);
      }
      printf("\n%d\n", y);
```

Answer. This prints out

```
7
0 2 4 6 8 10 12 14 16 18
7
```

```
4.   char str[10] = "strange";
      int len = strlen(str);
      printf("%d\n", len);

      for (int i = 2; i < len - 2; i++) {
          printf("%c", str[len - i]);
      }
```

Answer. This prints out

```
7
gna
```

```
5.   int thing1(int i) {
      static int j = 5;
      j += i;
      i = j * 2;
      return i;
  }

  int thing2(int x, int y) {
      int z = thing1(x) + thing1(y) + x + y;
      return z;
  }

  int main() {
      int a = 2;
      int b = 5;
      printf("a=%d, b=%d, thing2(a, b)=%d", a, b, thing2(a, b));
      return 0;
  }
```

Answer. This prints out

```
a=2, b=5, thing2(a, b)=45
```

```
6.   int i;
      for (i = 0; i < 100; i++) {
          while (i % 10) {
              i++;
          }
          printf("%d ", i);
      }
```

Answer. This prints out

```
0 10 20 30 40 50 60 70 80 90 100
```

Code Reading

```
7.  /* takes an array and an integer specifying the length of the array */
    double arrayFunc(int array[], int length) {
        if (length < 1) {
            return 0;
        }

        double s = 0;
        for (int i = 0; i < length; i++) {
            s += array[i];
        }
        return s/length;
    }
```

Answer. This returns the double precision average of all elements in the array. Returns 0 if invalid length.

```
8.  for (int n = 2; n < 100; n++) {
        int p = 1;
        for (int d = 2; p && (d < n); d++) {
            if (n % d == 0) {
                p = 0;
            }
        }
        if (p) {
            printf("%d ", n);
        }
    }
```

Answer. This prints all prime numbers less than 100 and beginning with 2.

Code Writing

9. Write a function that takes a character array (assume it is a properly formatted C-style string), and prints the complete string to stdout backwards. Be sure to print all (and only) the characters of the string.

```
void backstr(char string[]) {
    int len = strlen(string);

    for (int i = len - 1; i >= 0; i--) {
        printf("%c", string[i]);
    }

    printf("\n");
}
```

10. Write a loop that allows a user to enter as many integers as she wants, and then prints out the sum of those numbers (no arrays should be required). Use `scanf()` for input (remember that `scanf()` returns the number of things it successfully read, so you can just read numbers until `scanf()` fails to get an integer).

```
int main() {  
  
    int sum = 0;  
    int cur;  
  
    printf("Enter integers:\n");  
    int scan = scanf("%d", &cur);  
  
    while (scan) {  
        sum += cur;  
  
        scan = scanf("%d", &cur);  
    }  
  
    printf("\nSum: %d\n", sum);  
  
    return 0;  
}
```

Explanation

11. Explain why global-scope variables are considered to be poor practice.

Answer. They make debugging difficult because if multiple functions operate on a global-scope variable, it can be hard to find which one is causing the problem.

12. Explain why you need to specify `.c` files but not `.h` files on the command line when you call `gcc`.

Answer. `.h` files are specified at the header of each `.c` file, so they don't need to be specified again.

13. Explain why the C language is commonly used for writing operating systems, and why it is used less often for small, day-to-day programming tasks.

Answer. C can directly work on the hardware, which makes it extremely fast - good for operating systems. It isn't used as often for small tasks because it doesn't have much built in functionality and direct communication with the hardware can be unsafe.