



Part I: Pointers

a. Draw pictures, predict output for:

```

int main()
{
    int m, n;
    int *p1, *p2;

    p1 = &m;
    p2 = &n;
    *p1 = 3;
    *p2 = 5;
    cout << m << " " << n << endl;
}

```

b. Draw pictures, predict output for:

```

int main()
{
    int m, n;
    int *p1, *p2;

    p1 = &m;
    p2 = &n;
    *p1 = 8;
    *p2 = (*p1)++;
    *p2 += *p2;

    cout << m << " " << n << " "
         << *p1 << " " << *p2 << endl;
}

```

c. Draw pictures, predict output for:

```

int f(int*);

int main()
{
    int m = 6, n;

    n = f( &m );
    m = f( &n );
    cout << m << " " << n << endl;
}

int f(int* z)
{
    return 3 + *z ;
}

```

d. Draw pictures, predict output for:

```

int f(int*);

int main()
{
    int m = 5, n;

```

```

n = f( &m );

```

```

    cout << m << " " << n << endl;
}

int f(int* z)
{
    if ( *z <= 0 )
        return *z + 1;
    *z -= 2;
    return f( z );
}

```

e. Draw pictures, predict output for:

```

struct Person {
    string name;
    int age;
};

int main()
{
    Person x = { "Lee", 19 };
    Person y = { "Jan", 20 };
    Person *p, *q;
    p = &x;
    q = &y;
    (*p).age++;
    (*q).age += 3;
    x.name = (*q).name ;
    (*q).name = "Rob";
    cout << x.name << " " << x.age << endl
         << y.name << " " << y.age << endl;
}

```

f. Draw pictures, predict output for:

```

struct Day {
    int hitemp, lotemp;
};

int main()
{
    Day week[7];
    Day *t = &week[0];
    for(int i=0; i<7; i++){
        (*t).hitemp = 100 + i;
        week[i].lotemp = week[i].hitemp - 10;
        t++;
    }
    for(int i=0; i<7; i++){
        cout << "Day " << i << " "
             << week[i].lotemp << " "
             << week[i].hitemp << endl;
    }
}

```

Part II: Recursion

g. Predict output for:

```

void p(int);

int main()
{
    p(5);
    return 0;
}

void p(int a)
{
    if ( a < 10 ){
        cout << a << endl;
        p(a + 2);
    }
}

```

h. Predict output for:

```

string m(string);

int main()
{
    string v;

    v = m("hi");
    cout << v << endl;
    return 0;
}

string m(string s)
{
    if ( s.length() >= 6 ){
        return s;
    }
    s = m(s + '!');
    return s;
}

```

i. Predict output for:

```

int g(int[]);

int main()
{
    int a[] = { 1, 2, 4, 8, 16, 32 };
    int s = g(a, 5);
    cout << s << endl;
    return 0;
}

int g(int x[], int p)
{
    int s ;

    if ( p < 0 )
        return 0;
    s = x[p] + g(x, p-1);
    return s;
}

```

Part III: Char Arrays

j. Predict output for:

```

int main()
{
    string s1 = "quince tree";
    string s2 = "";
    char c;
    int len = s1.length();

    for(int i=0; i<len; i++){
        c = s1[i];
        if ( c != 'a' && c != 'e' && c != 'i'
            && c != 'o' && c != 'u' )
            s2 += c;
    }
    cout << s2 << endl;
}

```

k. Write a function that takes two strings as arguments and ‘zipper’ them together the way a clothing zipper interleaves alternate teeth. For example, if you zipper together "summer" and "picnic" you get "spuimcmneirc".

The signature of the function is:

```
string str_zipper(string s, string t)
```

l. The C library has a function called `isalpha(c)` that tells you if its argument is an alphabetic character. It returns a zero value if not, and it returns a non-zero value if the character is a letter of the alphabet.

Write a function that takes a string as an argument and returns the result of removing all non-letters from the original string. For example, if the argument is "Comp 11, Room 206", the return value is "CompRoom".

m. Write a function that reverses the zipper operation in the earlier problem. The function takes one string as an argument and returns the two strings as members of a struct that consists of two strings. The return type is:

```

struct StringPair {
    string s1;
    string s2;
};

```