

# Classes: A First Look

```
#include <iostream.h>
```

```
#define SIZE 10
```

```
// Declare a stack class for characters
```

```
class stack {
```

```
    char stck[SIZE]; // holds the stack
```

```
    int tos;          // index of top-of-stack
```

```
public:
```

```
    void init();          // initialize stack
```

```
    void push(char ch); // push character on stack
```

```
    char pop();          // pop character from stack
```

```
}
```

**// Initialize the stack**

```
void stack::init() { tos = 0; }
```

**// Push a character.**

```
void stack::push(char ch) {  
    if (tos==SIZE) { cout << "Stack if full"; return; }  
    stck[tos] = ch;  
    tos++; }
```

**// Pop a character**

```
char stack::pop() {  
    if (tos==0) { cout << "Stack is empty";  
                return 0; // return null on empty stack  
            }  
    tos--; return stck[tos]; }
```

```
main() {  
    stack s1, s2; // create two stacks  
    int i;  
    // initialize the stacks  
    s1.init();  
    s2.init();  
  
    s1.push('a');      s2.push('x');  
    s1.push('b');      s2.push('y');  
    s1.push('c');      s2.push('z');  
  
    for (i=0; i<3; i++) cout << "Pop s1: " << s1.pop() << "\n";  
    for (i=0; i<3; i++) cout << "Pop s2: " << s2.pop() << "\n";  
  
    return 0;  
}
```

This page intentionally left blank



# HW #3 (Coins)

- **Discussion:** Any monetary<sub>貨幣</sub> value in dollars and cents can be represented by an equivalent number of coins in the denomination<sub>面值</sub> of quarters<sub>25分硬幣</sub>, dimes<sub>10分</sub>, nickels<sub>5分</sub>, and pennies<sub>1分</sub> (some of which may be 0). For example, \$2.47 is equivalent to 9 quarters, 2 dimes and 2 pennies, or 247 pennies, or 24 dimes and 7 pennies, etc.
- **Problem:** Write a class (using **C++**, or **Java** or **Python** if you prefer) called **Coins** that represents a monetary value in dollars and cents as well as an equivalent number of  $q$  quarters,  $d$  dimes,  $n$  nickels, and  $p$  pennies, where  $q, d, n, p \geq 0$ . The class will compute the equivalent  $q, d, n$ , and  $p$  according to the algorithm described below.

# HW #3 (2)

- **Input:** Two monetary values in dollars and cents in the format  $x.y_1y_2$ . The two values will be input from the **command line**. Notice that any of the digits may be 0. For example, 0.07, 2.30, 7.03, 4.00, etc, are all valid inputs.
- **Output:**
  1. The two values in dollars and cents format ( $\$x.y_1y_2$ ) and their equivalent in  $q$ ,  $d$ ,  $n$ , and  $p$  values computed by the algorithm described in step 3 below.
  2. The format of the outputs should be as follows.  
 $\$2.33 = 9$  quarters, 0 dimes, 1 nickels, 3 pennies  
 $\$4.10 = 16$  quarters, 1 dimes, 0 nickels, 0 pennies

# HW #3 (3)

- **Minimum class requirements:**

1. At least one constructor that takes a double representing a dollars and cents value.
2. A member function to extract the dollars value and the cents value of the input and save them as integers. This function should be called from the constructor.
3. A member function to compute and save  $q$ ,  $d$ ,  $n$ , and  $p$ . This function should be called from the constructor.  
Notice that this function must be computed  $q$ ,  $d$ ,  $n$ , and  $p$  in that order, and that the maximum possible value must be computed for each from what currently remains after the predecessor is computed ( $q$  has no predecessor).

# HW #3 (4)

4. **Constant** members that do the following (make all of these implicitly **inlined**).
  - a. Return the dollar value.
  - b. Return  $q$ ,  $d$ ,  $n$ , and  $p$  as separate members.
  - c. A display function to display the equivalent values of  $q$ ,  $d$ ,  $n$ , and  $p$  as computed by step 3 above. Notice that this function displays all values.
- A friend operator  $<<$  to display the dollars and cents value in the format  $\$x.y_1y_2$ . For example, \$2.33. 不用做



# HW #3 (5)

- **Driver requirements:**

1. Read the two monetary values from the command line in `argv[1]`.
2. Convert `argv[1]` to double form and use the resulting value to instantiate the **Coins** class. Notice the type of `argv[1]` is `char*` which must be converted to float or double.
3. Produce an annotated display as described in the output requirements.