Introduction to Computer Science I Using C++
CSC 1253
Louisiana State University

Perfect Numbers—Programming Assignment 4 A *perfect number* is an integer $p>1$ such that the sum of the positive divisors of p equals p. For example, 6 is a *perfect number* since 1+2+3 = 6; 28 is a *perfect number* since 1+2+4+7+14 = 28.

Write a complete C++ program to write to the file `perfect.out` all *perfect number*s between 1 and a specified named integer constant `LIMIT`. For the purposes of the assignment, set the value of `LIMIT` to 10000.

The program is to define and use the following two functions:

```
int sum_of_positive_divisors(int number)
```

that calculates and returns the sum of all positive divisors of the integer argument `number`; candidate divisors of `number` range from 1 to `number / 2`. `sum_of_positive_divisors()` will only be called by `is_perfect()`.

```
bool is_perfect(int number)
```

is a predicate (a boolean-valued function) that returns `true` iff the integer argument `number` is a *perfect number* and `false` otherwise. `is_perfect()` will call `sum_of_positive_divisors()`; `is_perfect()` will only be called by `main()`.

Making Change—Programming Assignment 5 Write a complete C++ program to do the following:

1. Read a single input record from the text file `transactions.dat` containing the initial number of each coin and currency type in a till from which change for purchases in a shop is to be made. Echo this information to the text file `change.out`.
2. Read an arbitrary number of input records from the same file, each containing shop transactions consisting of a purchase cost followed by the

amount tendered by the customer to pay for the purchase. For each transaction, print to the text file `change.out` the following information:

(a) the cost

(b) the payment

(c) the change and the number of each coin and banknote in the change

(d) the resulting till contents (the number of each respective coin or currency type remaining in the till after the transaction is completed). As a simplifying assumption the total amount tendered by the customer for the purchase goes into a lock box, not into the till. This implies that the till contants will either remain the same (no change for the transaction) or decrease by the number of each coin or currency provided in the change.

If the input file `transactions.dat` contains:

```
5 5 10 20 40 50 40 50
2.83 3.00
2.34 20.00
2.53 100.00
```

Then the output to `change.out` will be:

```
Initial Till Contents:
5 twenties 5 tens 10 fives 20 dollars 40 quarters 50 dimes 40 nickels 50 pennies

Cost =    2.83
Payment =    3.00
Change =    0.17, consisting of:
1 dime 1 nickel 2 pennies
Till Contents:
5 twenties 5 tens 10 fives 20 dollars 40 quarters 49 dimes 39 nickels 48 pennies

Cost =    2.34
Payment =   20.00
Change =   17.66, consisting of:
1 tenner 1 fiver 2 dollars 2 quarters 1 dime 1 nickel 1 penny
Till Contents:
5 twenties 4 tens 9 fives 18 dollars 38 quarters 48 dimes 38 nickels 47 pennies

Cost =    2.53
Payment =  100.00
Change =   97.47, consisting of:
4 twenties 1 tenner 1 fiver 2 dollars 1 quarter 2 dimes 2 pennies
Till Contents:
1 twenty 3 tens 8 fives 16 dollars 37 quarters 46 dimes 38 nickels 45 pennies
```

The following function is required in your solution:

```
void make_change(float cost, float payment, float &change,
                 int &twentiesInTill, int &tensInTill, int &fivesInTill,
                 int &dollarsInTill, int &quartersInTill, int &dimesInTill,
                 int &nickelsInTill, int &penniesInTill, int &twentiesInChange,
                 int &tensInChange, int &fivesInChange, int &dollarsInChange,
                 int &quartersInChange, int &dimesInChange,
                 int &nickelsInChange, int &penniesInChange)
/*
 * For a purchase 'cost' and payment tendered in 'payment', makeChange
 * calculates and returns 'change' and the number of each individual
 * banknote or coin contained in that change from twenty dollar
 * banknotes to pennies as well as the resulting quantity for each
 * coin and currency remaining in the till.
 *
 * pure import parameters:
 *    cost - the cost of the item purchased
 *    payment - the payment tendered for the purchased item
 * pure export parameters:
 *    change - difference between the cost and payment
 *    twentiesInChange - the number of twenty dollar banknotes in change
 *    tensInChange - the number of ten dollar banknotes in change
 *    fivesInChange - the number of five dollar banknotes in change
 *    dollarsInChange - the number of one dollar banknotes or coins in change
 *    quartersInChange - the number of quarter dollar coins in change
 *    dimesInChange - the number of dime coins in change
 *    nickelsInChange - the number of nickel coins in change
 *    penniesInChange - the number of penny coins in change
 * import/export parameters
 *   (Each of the following initially contain as import parameters the
 *   number of the respective coin or currency in the till prior to the
 *   transaction.  As export parameters, they each contain the number of
 *   the respective coin or currency in the till following the transaction.
 *   Because of the problem simplification, the amount remaining of each
 *   quantity is simply what was there prior to the transaction minus the
 *   number given out in change.)
 *    twentiesInTill - the number of twenty dollar banknotes in the till
 *    tensInTill - the number of ten dollar banknotes in the till
 *    fivesInTill - the number of five dollar banknotes in the till
 *    dollarsInTill - the number of one dollar banknotes or coins in the till
 *    quartersInTill - the number of quarter dollar coins in the till
 *    dimesInTill - the number of dime coins in the till
 *    nickelsInTill - the number of nickel coins in the till
 *    penniesInTill - the number of penny coins in the till
 *
 * If the change in pennies to be returned to the customer is 4605 (representing
 * $46.05 in change) and twentiesInTill is at least 2, then twentiesInChange
 * would be set to 2 (which would be subtracted from twentiesInTill) and the
 * change in pennies would become 605. However, if there were only 1
 * twenty-dollar note in the till, that would be given in the change instead.
 * twentiesInTill would then be zero; twentiesInChange would be set to 1 and
```

```
 * the change in pennies would become 2605.

 * The total number of banknotes and coins provided in the change is minimised.
 */
{
}
```

An important hint here is that the change in pennies has to be assigned the following after the value of `change` is calculated:

```
changeInPennies = static_cast<int>(change * 100.0 + 0.001);
```

This compensates for any round-off error.

The file `transactions.dat` and a file called `makechange` containing all the lines provided above for the function `makeChange` (all you need to do is add the actual statements within to complete its definition) can be each copied from "the usual places". If you plan to do all your work remotely on the `classes` server, the best way to start on the assignment is to copy the `makechange` file as the starting contents of your program; simply create the directory `prog5`, change to that directory and then:

classes> `cp ~cs1253_bla/makechange .`

and paste or insert the file contents into your source file `changer.cc`.

or fetch the files from the website to your machine through

`http://www.haruspex.net/csc1253/makechange`

and

`http://www.haruspex.net/csc1253/transactions.dat`

Now simply use an editor to add to what you copied to complete the program.

Assignment #4 is due no later than 12.00 Noon on Thursday, 14^{th} October.

Assignment #5 is due no later than 12.00 Noon on Thursday, 21^{st} October.

Each assignment is to be submitted electronically. The *perfect numbers* (4^{th}) assignment will be done in a subdirectory of the student's home directory called `prog4`, contained in a file called `perfect.cc`, and submitted using the command

classes> `~cs1253_bla/bin/p_copy 4`

The *making change* (5^{th}) assignment will be done in a subdirectory of the student's home directory called `prog5`, contained in a file called `changer.cc`, and submitted using the command

classes> `~cs1253_bla/bin/p_copy 5`