CSC180 Introduction to Computer Programming
Assignment 2

Due: 11:59 PM, October 27, 2008

1 Problem description

Improve the receipt machine in the assignment 1. The program reads transaction time and purchases from standard input. Assume that standard input is always valid. Hint: these library functions might be useful: `strcpy` (or `strncpy`), `strcmp` (or `strncmp`), and `sscanf`. This assignment is worth 4% toward your final grade.

The receipt must have exactly the same format as shown below:

```
        Welcome to U of T Bookstore
        Koffler Student Services Bldg.
        214 College St.
        Toronto, ON M5T 3A1
        Phone: 416.640.7900
        GST # R132094343

                Sale
Cashier: 263            10/10/08 12:00

        11 2b                   $21.78
         2 09                   $21.64
        12 DD                   $12.12

        Subtotal:               $55.54
Tax:
 GST:                           $2.17
 PST:                           $1.73

                                ------
        Total:                  $59.44

 DEBIT CARD                     $60.00
 12345678********

Change due:                     $0.56

        Receipt required for all refunds
```

The above receipt was generated using the following function calls and the standard input under the second point in Section 2:

```
init(263);
purchase();
cancel(``A1'', 3);
print(60.0, ``debit'', ``1234567887654321'');
```

We provide `receipt.h`; its contents are listed below:

```
/* receipt.h */
#define NUM     99   /* max number of distinct items */
#define SIZE    15   /* size of time string */
#define GST   0.05   /* GST tax rate */
#define PST   0.08   /* PST tax rate */
#define BANNER ``\tWelcome to U of T Bookstore\n\tKoffler Student Services Bldg.\n'' \
               ``\t214 College St.\n\tToronto, ON M5T 3A1\n\tPhone: 416.640.7900\n'' \
               ``\tGST # R132094343\n\n\t\tSale\n''

/* from getline.c */
int getline(char s[], int lim);

/* from receipt.c */
void init(int id);
void purchase(void);
void cancel(char *s, int q);
void tax(char t[], int q[], double p[], int n, double *gst, double *pst);
double subtotal(int q[], double p[], int n);
double total(double t, double gst, double pst);
double change_due(double a);
void print(double a, char *opt, char *cno);
```

We also provide `getline.c`; its contents are listed below:

```
/* getline.c */
#include <stdio.h>

/* get line into s, return length */
int getline(char s[], int lim)
{
  int c, i;

  i = 0;
  while (--lim > 0 && (c=getchar()) != EOF && c != '\n')
    s[i++] = c;
  if (c == '\n')
    s[i++] = c;
  s[i] = '\0';

  return i;
}
```

You must use `receipt.h` and `getline.c` in your program.

You write `receipt.c`, in which you must use the following static external variables:

```
/* static external variables */
static double price[NUM];  /* item price */
static int quantity[NUM];  /* item quantity */
static char tax_type[NUM]; /* tax on item */
static char name[NUM][2];  /* item name */
static char time[SIZE];    /* transaction time */
static int num;            /* number of distinct items purchased */
static int cashier;        /* cashier id */
```

You must define the functions in Section 2 using exact function prototypes.

2 Function prototypes

- `void init(int id);`

  Initialize the receipt machine for a transaction. Each transaction begins with calling `init`. This function initializes the external variables. It sets `cashier` to `id`, `time` to ‘‘`00/00/00 00:00`’’, `num` to 0, all elements in `price` to 0, all elements in `quantity` to 0, all elements in `tax_type` to `'U'`, and all elements in `name` to `'\0'`.

- `void purchase(void);`

  This function changes the values of the static external variables using the information read from standard input. The above receipt was generated using the following standard input:

  ```
  10/10/08 12:00
  A1:2.31:1:P
  2b:1.98:2:g
  2b:1.98:9:g
  09:10.82:2:B
  DD:1.01:10:N
  DD:1.01:2:N
  A1:2.31:2:P
  ```

  In UNIX, typing Ctrl-D indicates the end of standard input, thus makes the function `getchar` return `EOF`.

  The standard input contains only transaction time and purchases. The first line is transaction time, with the format `dd/mm/yy hh:mm`. Each line thereafter is a purchase, consisting of four fields: `item name`, `price`, `quantity`, and `tax type`. Each field is present, and neighboring fields are separated by a colon. There are no whitespaces around colons. `item name` contains two and only two characters, chosen from `a-z`, `A-Z`, and

0-9. price is only allowed to be an integral number of cents: .22, 10, 2.1, and 2.12 are examples of valid prices; −1.2 and 1.212 are examples of invalid prices. quantity is a non-negative integer. Acceptable values for tax type are described in the assignment 1. Let's take the second line, A1:2.31:1:P, for example. A1 is item name. An item name is case sensitive; thus A1 is different from a1. 2.31 is price. 1 is quantity. P is tax type.

The standard input may be empty, or contains only transaction time. Ensure that purchase still works in these situations.

Assume that the number of distinct items does not exceed NUM. In the above standard input, the distinct items are A1, 2b, 09, and DD; thus the number of distinct items is 4. The distinct items are stored and printed in the order they first appear. Only distinct items are stored in the arrays. For example, A1 is stored in name[0][0] and name[0][1], 2.31 in price[0], total quantity 3 in quantity[0], and 'P' in tax_type[0]. An item has one and only one tax type, represented by a unique character.

purchase will be called at most once in a transaction.

Note: you are not allowed to use scanf or fscanf. You must use getline and sscanf when defining this function.

- `void cancel(char *s, int q);`

  Cancel q items with item name given by s.

  A cancellation has effects if and only if at least one item s has been purchased, and q is not greater than the number of item s purchased; otherwise, cancel still works but has no effects on the static external variables.

  If item s has been totally cancelled, its name does not show on the receipt: the receipt must not have a line as follows:

  ```
  0 A1                    $0.00
  ```

- `void tax(char t[], int q[], double p[], int n, double *gst, double *pst);`

  Compute taxes imposed on effective purchases. Purchases that are not cancelled are considered effective. t contains tax types, q contains quantities, p contains prices, and n is the number of distinct items. gst and pst are pointers to double.

- `double subtotal(int q[], double p[], int n);`

  Return total cost (before tax). The meanings of q, p, and n are the same as those in the function tax.

- `double total(double t, double gst, double pst);`

  Return total cost (after tax, if any). t is subtotal amount. gst and pst are GST tax and PST tax, respectively.

- `double change_due(double a);`

  Return change due given payment `a`.

- `void print(double a, char *opt, char *cno);`

  Given payment `a`, payment option `opt`, and card number `cno`, print a U of T Bookstore receipt. Acceptable payment options are ``debit'' (Debit Card), ``mc'' (Master Card), ``vc'' (Visa Card), and ``cash''. Each transaction allows only one payment option. The card number is a 16-digit string. The last 8 digits are replaced with 8 *'s when printed out on the receipt. Note that if the payment option is ``cash'', `cno` is an empty string, and the card number line on the receipt does not exist.

  `print` is the only function in this assignment that prints information on standard output. All other functions must not print anything.

3 Receipt format

Failure to satisfy the following requirements will result in considerable mark deductions.

- The receipt must have exactly the same format as the one shown above, except that the numbers, the transaction time, the item list, the payment option, or the card number (if any), may vary. Quantities must be integers greater than 0, with no decimal point nor fraction part. Money must be preceded by `$`, and must have two and only two digits after decimal point.

- You must use the macro `BANNER` to print the receipt header. You must use the following format strings when printing corresponding lines:

```
``Cashier: %d\t\t%s\n''
``\t%2d %c%c \t\t\t$%.2f\n''   (an item in the item list)
``\tSubtotal: \t\t$%.2f\n''
``Tax:\n'';
`` GST: \t\t\t\t$%.2f\n''
`` PST: \t\t\t\t$%.2f\n''
``\t\t\t\t------\n''
``\tTotal: \t\t\t$%.2f\n''
`` DEBIT CARD \t\t\t$%.2f\n''
`` MASTER CARD \t\t\t$%.2f\n''
`` VISA CARD \t\t\t$%.2f\n''
`` CASH \t\t\t$%.2f\n''
``Change due: \t\t\t$%.2f\n''
``\tReceipt required for all refunds\n''
```

- The receipt must not have duplicated item names. An example of duplicated item names is given below:

```
        2 2b                    $3.96
        9 2b                    $17.82
```

- If standard input is empty, the receipt has the following format:

```
        Welcome to U of T Bookstore
        Koffler Student Services Bldg.
        214 College St.
        Toronto, ON M5T 3A1
        Phone: 416.640.7900
        GST # R132094343

                Sale
Cashier: 263            00/00/00 00:00

        Subtotal:               $0.00
Tax:
 GST:                           $0.00
 PST:                           $0.00

                                ------
        Total:                  $0.00

 DEBIT CARD                     $60.00
 12345678********

Change due:                     $60.00

        Receipt required for all refunds
```

  The above receipt was generated using exactly the same sequence of function calls in Section 1.

- If standard input contains only transaction time, the time must be properly printed on the receipt.
- Payment option ‘‘`debit`’’ prints `DEBIT CARD`, as shown above. ‘‘`mc`’’ prints `MASTER CARD`. ‘‘`vc`’’ prints `VISA CARD`. ‘‘`cash`’’ prints `CASH`.
- You must make the space between lines exactly the same as the receipt examples provided.

4 Submission

- You must submit one and only one file, `receipt.c`. Include the following line near the top of `receipt.c`:

```
#include ‘‘receipt.h’’
```

`receipt.c` must not contain `main` function, nor other library header files except `stdio.h` and `string.h`. You may test the functions defined in `receipt.c` using another file, e.g., `main.c`. `main.c` may look like this:

```
#include <stdio.h>

#include ``receipt.h''

int main()
{
  /* test functions from receipt.c */
  return 0;
}
```

- Insert the following comment on the top of `receipt.c`:

```
/*
 * receipt.c -- description of this file, if any
 *
 * $name, 012345678$
 */
```

  Replace `name` with your name, and replace `012345678` with your student number.

- Comment and indent `receipt.c` properly
- Follow the coding style posted on Discussion Board (the section General)
- Compilation: compile the program using the following command:

```
gcc -Wall getline.c receipt.c main.c -o receipt
```

- To save typing, save standard input in the file `input.txt` and redirect it to the program `receipt`:

```
./receipt < input.txt
```

- Submission methods: `submitcsc180f 2 receipt.c`