DSA 5005 Data Structures – Summer 2017 – Programming Project 1
Due June 30, 2017 – 11:59 PM

Objectives

1. [10 pts] Use basic input/output in C++ (cin and cout).
2. [10 pts] Use arrays in C/C++.
3. [10 pts] Creating C++ classes.
4. [50 pts] Design and implement the program *according to the project description*. 50 pts are distributed as follows:
   a. [10 pts] Read the input file using *redirected input*; print out each line read from the file
   b. [10 pts] Create appropriate item object based on the input and fill all the fields
   c. [10 pts] Add items to the Bag (Array of Objects described below)
   d. [10 pts] Print out the information of a specific item by going through the Bag
   e. [10 pts] Summary information about the items in the Bag, including the number of items in each item category, total price for each item category, and total value of all items in the Bag
5. [20 pts] Document your project thoroughly as the examples in the textbook. This includes but not limited to header comments for all classes/methods, explanatory comments for each section of code, meaningful variable and method names, and consistent indentation.

Project Description

An item can be pens, printer, or a safe box. Each item has unique characteristics (Packet Size for Pens, Tray Capacity for Printer, and Electronic Lock for Safe Box). See that table below.

In this project, you will create a class called Item that has common attributes 1-10 given below as its fields. Note that the interpretation of attribute 10 is different even they the data type is the same (bool for Boolean). The Item class will also have a field named Code (1 for Pens, 2 for Printer, and 3 for Safe Box. In addition to the Item class you will also create a class for Pens, Printer, and Safe Box. These class will inherit the Item class and have its own field (Packet Size for Pens, Tray Capacity for Printer, and Electronic Lock for Safe Box). You must write for each class, a) empty constructor, b) non-empty constructor that initializes all the fields, c) appropriate setter and getter methods, and d) display method that prints the **values of all the fields with the interpretation given in the table**.

Note that for data types, attributes 1-3 and 9, are int, 4-8 is double, 10 is bool (for Boolean), and 11 is int for Pens and Printer and bool for Safe Box.

Attribute	Pens	Printer	Safe Box
1	Code (1 for Pens)	Code (2 for Printer)	Code (3 for Safe Box)
2	Color (1 – Red; 2 – Blue; 3 – Green, 4 - Black)	Color (1 – Black, 2- White)	Color (1 – Black, 2 – Red)
3	Brand (1 – Pilot, 2 – Paper mate, 3 – Uni-ball, 4 – Sharpie)	Brand (1 – Dell, 2 – HP, 3 – Cannon, 4 – Brother)	Brand (1 – SentrySafe, 2 – First Alert, 3 – Honeywell)
4	Height	Height	Height
5	Length	Length	Length
6	Width	Width	Width
7	Weight	Weight	Weight
8	Price	Price	Price
9	Type (Ball Point – 1; Roller Ball – 2; Fountain – 3; Calligraphy -4)	Type (1 – All-In-One; 2 – Laser; 3 – InkJet)	Type (1 – Water Proof; 2 – Wall Mountable, 3 – Desktop)
10	Assorted Ink (0 – No, 1 – Yes)	Network (0 – No, 1 – Yes)	Fire Resistant (0 – No, 1 – Yes)
11	Packet Size	Tray Capacity	Electronic Lock (0 – No, 1 – Yes)

A Bag is declared as follows:

Item* Bag[1000]; //We can assume that we can create up to 1000 item pointers.

Assume that the first line of input (in the text file described below is as follows):

```
AddItem 5 1 2 3 7.75 7.62 0.69 0.025 4.97  2   0  10
   1    2 3 4 5  6    7     8    9    10  11  12  13
```

The 1 above corresponds to the command (we will only have AddItem), 2 corresponds to the quantity of this Item (here 5 means that we have 5 sets of pens), and 3 corresponds to the Code type (here it is 1 indicating Pens). Input positions 3-13 correspond to attributes 1-11 of the Item (in this case, Pens).

Upon successfully reading the above line of input you are required to the create the corresponding object (in this case it is a Pens Object) and set the appropriate fields with the values read. The number of Pens objects is equal to the quantity given in position 2 in the input (in the above example it is 5). Each of the Pens objects are stored in successive array positions in the Bag array given above. You will use the following to create each Pen Object and store it in the Bag array. For example, for the above input:

```
bagPos = 0; //assume this is the first item that we added
for (int i=0; i < 5; i++)
       Bag[bagPos+i] = new Pens(1, 2, 3, 7.75, 7.62, 0.69, 0.025, 4.97, 2, 0, 10)
```

After successfully reading all the input and storing in the Bag array, you can process the Bag array by appropriately type casting as follows to print the contents of the array as follows:

```
//Let noItems be the number of Item objects stored (it is the number Item object pointers)
//stored in the Bag
for (int i=0; i < noItems; i++) {
       switch ((*Bag[i]).Code) {
          case 1: {//Pens
                      (static_cast<Pens*>(Bag[i]))->display();
                      break;
                  }
          case 2: {//Printers
                      (static_cast<Printer*>(Bag[i]))->display();
                      break;
                  }
          case 3: {//Safe Box
                      (static_cast<SafeBox*>(Bag[i]))->display();
                      break;
                  }
          default: cout << "I do not recognize this Item in the Bag" << endl;
       }
}
```

You can use a similar looping structure to get the summary information about the items in the Bag, including the number of items in each item category (Pens, Printer, and Safe Box), total price for each item category, and total value of all items in the Bag.

Input File

The input file will consist an unknown number of lines and each line will be a command with the parameters of that command (if any). A short input file is provided below and a longer one will be posted to D2L later this week.

```
AddItem 5 1 2 3 7.75 7.62 0.69 0.025 4.97 2 0 10
AddItem 1 2 1 1 17.97 21.51 7.69 4.05 200.97 2 0 2000
AddItem 1 1 1 2 7.75 7.62 0.69 0.025 9.89 1 1 10
AddItem 1 3 2 3 14.00 13.00 19.75 150.95 112.87 3 1 1
```

You will implement the C++ classes as described in this section, as well as the `main()` method which will read and execute the commands from standard I/O.

You will implement the main() method as the code provided below.

```
#include <iostream>

// define and implement the required classes here,
// or in separate files, you will need to include the headers
/  for these classes in the latter case.

using namespace std;

int main()
{
   Item* Bag[256];
   Pens *onePen;
   Printer* onePrinter;
   SafeBox* oneSafeBox;
   // other local variables

   while( !cin.eof())  // while end of line is not reached
   {
      // read a line from input
      //cin >> command >> quantity >> code >> all other attributes
      //Based on code create the appropriate object and store it in the
      //Bag array as described previously
   }
   // Write code to display each item stored in the Bag array
   // Using code described previously
   // Print the summary statitics

   return 0;
}
```

Redirected Input

Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. To use redirected input in Visual Studio environment, follow these steps: After you have opened or created a new project, on the menu go to Project, Project Properties, expand Configuration Properties until you see Debugging, on the right you will see a set of options, and in the Command Arguments type "< **input filename**". The < sign is for redirected input and the **input filename** is the name of the input file (including the path if not in the working directory).

Constraints

1. In this project, the only header you will use is #include <iostream>.
2. None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.