Examen final ejemplo – Práctica

Parte I

1. Crear tres archivos
   - `ItemToPurchase.h` – Declaración de clase
   - `ItemToPurchase.cpp` – Definición de la clase
   - `Main.cpp` – función principal.

   Construir la clase ItemToPurchase con las siguientes especificaciones:
   - Constructor por defecto
   - Funciones miembro públicas (mutator & accessors)
     - `SetName()` & `GetName()` (2pts)
     - `SetPrice()` & `GetPrice()` (2pts)
     - `SetQuantity()` & `GetQuantity()` (2 pts)
   - Miembros privados
     - `string itemName` – Inicializa el constructor por defecto a “none”
     - `int itemPrice` – Inicializa el constructor por defecto a 0
     - `int itemQuantity` – Inicializa el constructor por defecto a 0
2. En `main()`, pida al usuario 2 items y cree dos objetos de la clase `ItemToPurchase`. Antes de pedir el segundo item, llame a `cin.ignore()` para permitir al usuario entrar un nuevo string. (2pts)

   Ex:

   ```
   Item 1
   Enter the item name:
   Chocolate Chips
   Enter the item price:
   3
   Enter the item quantity:
   1

   Item 2
   Enter the item name:
   Bottled Water
   Enter the item price:
   1
   Enter the item quantity:
   10
   ```

3. Sume el costo de los dos ítems y muestre el resultado del costo total (2 pts)

Ex:

```
TOTAL COST
Chocolate Chips 1 @ $3 = $3
Bottled Water 10 @ $1 = $10

Total: $13
```

Nota: Se realizarán 6 pruebas automáticas hasta este momento que sumarán un total de 10 puntos. Se usarán **funciones similares** a la siguiente para las pruebas. Para las partes I y II (la siguiente), cree carpetas diferentes para almacenar sus archivos respectivos.

```
bool testPassed() {

   ItemToPurchase item;
   item.SetName("Chocolate Chips");

   if(item.GetName() == "Chocolate Chips") {
      cout << "Item name \"Chocolate Chips\" correctly set and is accessible." << endl;
      return true;
   }
   else {
      cout << "Item name \"Chocolate Chips\" incorrectly set or is inaccessible." << endl;
      cout << "Incorrect name returned: " << item.GetName() << endl;
      return false;
   }
}
```

Parte II

Recuerde crear una nueva carpeta

4. Extienda la clase `ItemToPurchase` a las siguientes especificaciones:
   - Parametrice el constructor para asignar nombre, descripción del item, precio del item y cantidad del item (usar valores por defecto de 0). (1 pt)
   - Adicionar las siguentes funciones miembro:
     - `SetDescription()` mutator & `GetDescription()` accessor (2pts)
     - `PrintItemCost()` – Imprime el nombre del item seguido por la cantidad, precio y subtotal.
     - `PrintItemDescription()` – Imprime el nombre del item y la descripción
   - Miembros privados
     - `string itemDescription` – Inicializa el constructor por defecto a "`none`"

Ex. of PrintItemCost() output:

```
Bottled Water 10 @ $1 = $10
```

Ex. of PrintItemDescription() output:

```
Bottled Water: Deer Park, 12 oz.
```

5. Cree tres nuevos archivos
    - `ShoppingCart.h` – Declaración de clase
    - `ShoppingCart.cpp` – Definición de clase
    - `main.cpp` – función principal (Nota: contiene funcionabilidad diferente a la parte I)

    Construya la clase `ShoppingCart` con las siguientes especificaciones. Nota: algunas funciones pueden ser stubs (funciones vacías) inicialmente, serán implementadas en pasos siguientes.

    - Constructor por defecto
    - Parametrizar el constructor el cual toma el nombre del cliente y la fecha como parámetros
    - Miembros privados
        - `string custumerName` – Inicializa el constructor por defecto a "`none`"
        - `string currentDate` – Inicializa el constructor por defecto a "December 17, 2021"
        - `vector <itemToPurchase> cartItems`, esta estructura es una implementación de **arreglos dinámicos** en la librería estándar de C++, ver documentación y ejemplos de uso en https://www.geeksforgeeks.org/vector-in-cpp-stl/
    - Funciones miembro públicas
        - `GetCustomerName()` accessor (1pt)
        - `GetDate()` accessor (1 pt)
        - `AddItem()` – Adiciona un item al arreglo dinámico `cartItems`. Tiene un parámetro de tipo `ItemToPurchase`. No devuelve nada.
        - `RemoveItem()` – Remueve ítems del arreglo dinámico `cartItems`. Tiene un `string` (el nombre de item) como parámetro. No devuelve nada. Si no se encuentra el nombre del item, imprimirá "`Item not found in cart. Nothing removed`".
        - `ModifyItem()` – Modifica la descripción, precio y/o la cantidad de un item. Tiene un parámetro de tipo `ItemToPurchase`. No devuelve nada. Si el item se puede encontrar (por nombre) en el carrito, verifique si los parámetros tienen valores predeterminados para la descripción, precio y cantidad. Si no es así, modifique el item en el carrito. Si no se puede encontrar el item (por nombre) en el carrito, imprima este mensaje: "`Item not found in cart. Nothing modified`".
        - `GetNumItemsInCart()` (2 pts) – Devuelve la cantidad de ítems en el carro. No tiene parámetros.
        - `GetCostOfCart()` (2 pts) – Determina y devuelve el costo total de los ítems en el carrito. No tiene parámetros.
        - `PrintTotal()` – Imprime en número total de objetos en el carrito de compras. Si está vacío, imprimir el mensaje "`SHOPPING CART IS EMPTY`".
        - `PrintDescription()` – Imprime la descripción de cada item en el carrito de compras

Ex. of PrintTotal() output:

```
John Doe's Shopping Cart - February 1, 2016
Number of Items: 8

Nike Romaleos 2 @ $189 = $378
Chocolate Chips 5 @ $3 = $15
Powerbeats 2 Headphones 1 @ $128 = $128

Total: $521
```

Ex. of PrintDescriptions() output:

```
John Doe's Shopping Cart - February 1, 2016

Item Descriptions
Nike Romaleos: Volt color, Weightlifting shoes
Chocolate Chips: Semi-sweet
Powerbeats 2 Headphones: Bluetooth headphones
```

6. En main(), solicitar al usuario el nombre de un cliente y la fecha de hoy. Imprimir el nombre y la fecha. Cree un objeto de tipo `ShoppingCart`. (1 pt)

Ex:

```
Enter customer's name:
John Doe
Enter today's date:
February 1, 2016

Customer name: John Doe
Today's date: February 1, 2016
```

7. Implemente la función `PrintMenu()` en `main.cpp` para imprimir el siguiente menú de opciones que permiten manipular el carrito de compras. (1 pt)

Ex:

```
MENU
a - Add item to cart
d - Remove item from cart
c - Change item quantity
i - Output items' descriptions
o - Output shopping cart
q - Quit
```

8. Implemente la función `ExecuteMenu()` en `main.cpp` que toma 2 parámetros: un carácter que representa la elección del usuario y la referencia de un carrito de compras.

`ExecuteMenu()` realiza las opciones de menú que se describen a continuación, de acuerdo con la elección del usuario. (1 pt)

9. En `main()`, llame a `PrintMenu()` y pida que el usuario elija una opción del menú. Cada opción está representada por un solo carácter.

   Si se ingresa un carácter no válido, continúe solicitando una opción válida. Cuando se ingresa una opción válida, ejecute la opción llamando a `ExecuteMenu()`. Luego, imprima el menú y solicite una nueva opción. Continúe hasta que el usuario ingrese '`q`'. **Sugerencia**: implemente `Quit` antes de implementar otras opciones. (1 pt)

   Ex:

```
MENU
a - Add item to cart
d - Remove item from cart
c - Change item quantity
i - Output items' descriptions
o - Output shopping cart
q - Quit

Choose an option:
```

10. Implemente la opción “Output shopping cart” en ExecuteMenu(). (3 puntos)

    Ex:

```
OUTPUT SHOPPING CART
John Doe's Shopping Cart - February 1, 2016
Number of Items: 8

Nike Romaleos 2 @ $189 = $378
Chocolate Chips 5 @ $3 = $15
Powerbeats 2 Headphones 1 @ $128 = $128

Total: $521
```

11. Implemente la opción “Output item’s description” en ExecuteMenu (). (2 puntos)

    Ex:

```
OUTPUT ITEMS' DESCRIPTIONS
John Doe's Shopping Cart - February 1, 2016

Item Descriptions
Nike Romaleos: Volt color, Weightlifting shoes
Chocolate Chips: Semi-sweet
Powerbeats 2 Headphones: Bluetooth headphones
```

12. Implemente la opción de “Add item to cart “ en ExecuteMenu (). (3 puntos)

Ex:

```
ADD ITEM TO CART
Enter the item name:
Nike Romaleos
Enter the item description:
Volt color, Weightlifting shoes
Enter the item price:
189
Enter the item quantity:
2
```

13. Implemente la opción de “Remove item from cart “ en ExecuteMenu (). (4 puntos)

Ex:

```
REMOVE ITEM FROM CART
Enter name of item to remove:
Chocolate Chips
```

14. Implemente la opción de menú " Change item quantity" en ExecuteMenu (). **Sugerencia**: cree un nuevo objeto tipo ItemToPurchase y use los modificadores ItemToPurchase antes de usar la función ModifyItem(). (5 puntos)

Ex:

```
CHANGE ITEM QUANTITY
Enter the item name:
Nike Romaleos
Enter the new quantity:
3
```

Se realizarán 17 pruebas automáticas usando funciones similares a la siguiente que sumarán un total de 31 puntos adicionales.

```
bool testPassed() {

   ItemToPurchase item ("Bottled Water", "Deer Park, 12 oz.", 1, 10);

   if(item.GetName() == "Bottled Water" && item.GetDescription() == "Deer Park, 12 oz." &&
      item.GetPrice() == 1 && item.GetQuantity() == 10) {

      cout << "Item properly initialized." << endl;
      return true;
   }

   else {
      cout << "Item improperly initialized." << endl;
      cout << "GetName() returns " << item.GetName() << endl;
      cout << "GetDescription() returns " << item.GetDescription() << endl;
      cout << "GetPrice() returns " << item.GetPrice() << endl;
      cout << "GetQuantity() returns " << item.GetQuantity() << endl;
      return false;
   }
}
```

Bono: Permitir hacer sobrecarga de operadores +,- y * (Ejp. 3*objeto permite adicionar varias veces el mismo objeto), para poder operar tanto con ítems como con carritos de compra (10 pts)