CS 117 Programming II, Fall 2021 | Dr. Ghriga

Midterm Exam: 2 Hours 30 Minutes

October 25, 2021

Due Date: November 3, 2021 at 11:59PM

INSTRUCTIONS: Do all exercises. You are permitted to use all resources: CS117 zyBook or equivalent, class notes, practice programs, Programming environments Code::Blocks or Xcode, internet resources, etc. It is recommended that you avoid typing every program for answering the questions, be strategic in using the compiler. The following guidelines must be observed:

1) Do all exercises 1, 2, and 3 for total of 100 points.
2) The solution you submit must reflect your individual work. Cutting and pasting code from the internet or any other sources will not be graded. If plagiarism or cheating is established, a grade of zero for the exam will be assigned.
3) Do NOT submit programs. The solution you submit must clearly show the logic and correctness of your solution.
4) Your answers to all exercises should be clearly and neatly arranged in one single document with a title that carries your first name and last name.
5) All submissions must be postmarked by November 3, 2021 at 11:59PM or earlier.
6) Submit your document through Blackboard.

Exercise 1 [20 points]

What is the output of the following program? Explain your results and why certain elements of the vector did or did not change.

```
#include <iostream>
#include <vector>
using namespace std;

int mystery1(int a);
int mystery2(int& a);
void vprint(const vector<int>& v);
```

```
int main()
{
   vector<int> myVector;
   myVector.push_back(3);
   myVector.push_back(-5);
   myVector.push_back(5);
   vprint(myVector);
   cout << mystery1(myVector[1]) << endl;
   vprint (myVector);
   cout << mystery2(myVector[2]) << endl;
   vprint(myVector);

   return 0;
}

int mystery1(int a)
{
   a += 100;
   return a;
}

int mystery2(int & a)
{
   a += 1000;
   return a;
}

void vprint(const vector<int>& v)
{
   for (unsigned int i = 0; i < v.size(); i++)
      cout << v[i] << " ";
   cout << endl;
}
```

Exercise 2 [20 Points]

Write a function pattern that takes the size n of a square pattern as a parameter and displays an **n** by **n** pattern of numbers as shown below.

When n is 5, the pattern is:

1		1		1		1		1	
2		2		2		2		2	
3		3		3		3		3	
4		4		4		4		4	
5		5		5		5		5	

When n is 4, the pattern is:

1		1		1		1	
2		2		2		2	
3		3		3		3	
4		4		4		4	

Exercise 3 [60 points]

Consider the following class **ItemReview** and class **InventoryItem**:

```
class ItemReview
{
  public:
        // constructor
        ItemReview(): _starRating(0), _comment("") {}

       // get functions
       int getStarRating() const;
       string getReviewComment() const;

        // set functions
        void setStarRating(int k);
        void setReviewComment(string c);
```

```
        // destructor
        ~ItemReview() {}

  private:
         int _starRating;  // holds the number of stars, user rating, for a given item or product
         string _comment; // holds the user comment
};

class InventoryItem
{
  public:
        // constructor
        InventoryItem(): _ID(0), _description("no name"), _onHand(0) {}

       // get functions
       int getItemID() const;
       string getItemDescription() const;
       int getItemAvailability() const;

        // set functions
        void setItemID(int itemID);
        void setItemDescription(string desc);
        void setItemAvailability(int n);

       // other functions to manage the availability of the item
       void addItemShipmentReceived(int k); // adds k to the item availability
       void deductItemSales(int k); // deducts k from the item availability

        // other member functions to handle and process user reviews
        void addItemReview(ItemReview ir);  // adds ir to the list of user reviews
        void addItemReviews(vector<itemReview> v); // adds v to the list of user reviews
         vector<ItemReview>  getItemReviews() const;  // returns the vector of user reviews
         int getNumberOfReviews() const;    // returns the number of user reviews
         double getAverageStarRating() const; // returns the average of the user star ratings

       // nicely print all of the InventoryItem information, including summary of reviews
       // the reviews must be printed as follows:
       // Rating 5 followed by all review comments with 5 rating comments, one per line
```

```
        // Rating 4 followed by all review comments with 4 rating comments, one per line
        ….
        // Rating 1 followed by all review comments with 1 rating comments, one per line
        void display () const;

        // destructor
        ~InventoryItem () {}

  private:
        int _ID;  // holds ID of the inventory item
        string _description; // holds the description of the item
        int _onHand; // holds the number of available items
        vector<itemReview> _reviews;   // holds the user reviews
        // returns a vector of item reviews with rating r
        vector<itemReview> getSelectItemReviews(int r);
};
```

1. **[5 points]** Write the code for all get and set member functions for the class **ItemReview**:

```
// get functions
int getStarRating() const;
string getReviewComment() const;

// set functions
void setStarRating(int k);
void setReviewComment(string c);
```

   The user star ratings of an item range from 1(poor) to 5(excellent). If a rating larger than 5 is passed to function setStarRating(), the star rating is set to 5. Similarly, if a rating lower than 1 is passed, the star rating is set to 1. Hence, we will ensure that whenever we record a user star rating it is always in the valid range 1 to 5.

2. **[15 points]** Write the code for all member functions below:

```
// get functions
int getItemID() const;
string getItemDescription() const;
int getItemAvailability() const;
```

```
// set functions
void setItemID(int itemID);
void setItemDescription(string desc);
void setItemAvailability(int n);

// other functions to manage the availability of the item
void addItemShipmentReceived(int k); // adds k to the item availability
void deductItemSales(int k); // deducts k from the item availability
```

3. **[10 points]** Implement the following member functions which appear in the class Inventory Item interface:

```
void addItemReview(ItemReview ir);  // adds ir to the list of user reviews
void addItemReviews(vector<itemReview> v); // adds v to the list of user reviews
```

4. **[10 points]** Write the implementation of the following private utility function
   vector<itemReview> getSelectItemReviews(int r);

5. **[10 points]** Implement the member function **display**, which appears in the class **InventoryItem** interface. You must use the private utility function **getSelectItemReviews** for full credit.

6. **[10 points]** Add a copy constructor for class **ItemReview** and its implementation