CS 150 Lecture 16 Exercises

Complete each of the exercises below and upload to Canvas before the deadline.

A. Overloaded Output Operator

You can write output operators for any type. Write the overloaded output operator for the **Car** type. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of the output operator for the Car type* |

|  |
| --- |
| *Copy and paste a screenshot of the test results – No libcars.a for makefile compiliation* |

B. Introducing the vector Type

Follow along in class with your instructor in class to initialize five different **vector** objects. You'll find the comments in **vinit.cpp**. Test your code and paste the requested information in the text areas provided below.

|  |
| --- |
| *Copy and paste a screenshot of source code of vinit.cpp* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |

C. vector and Templates

Follow along in class with your instructor to write a function which will print any kind of **vector**. You'll find the comments in **vprint.cpp** (where you'll add your ID). The generic function will go inside **vprint.h** Test your code and paste the requested information in the text areas provided below.

|  |
| --- |
| *Copy and paste a screenshot of source code of print.h* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |

D. Writing a vector Output Operator

You can write output operators for any type. However, for the **vector** type, it is more useful to write one operator that works for **any kind** of **vector**. Do that by writing a **function template** which will generate the correct operators when used. Place your template in the header file **vecout.h**. Place only your name in **vecout.cpp**. I have included the other portions needed for testing in **libvecout.a**. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of the output operator for the vector type* |

|  |
| --- |
| *Copy and paste a screenshot of the test results – no libvecout.a* |

E. Fundamental Algorithms - Counting

Write the function **divisibleBy(v, n)** which counts the number of elements in the **vector<int>** **v** which are evenly divisible by **n**. Use the most appropriate loop for the problem. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of your source code here* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |

F. STL Iterators - Counting

Rewrite the function **divisibleBy(v, n)** which counts the number of elements in the **vector<int>** **v** which are evenly divisible by **n**. This time, implement the function using an iterator loop. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of your source code here* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |

G. STL Algorithms - Counting

Rewrite the function **divisibleBy(v, n)** which counts the number of elements in the **vector<int>** **v** which are evenly divisible by **n**. This time, implement the function using the appropriate STL algorithm. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of your source code here* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |

H. Functions Returning vector

Write the function **unique(v)** which returns a copy of the **vector<int>** **v** with all of the duplicates removed. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of your source code here* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |

I. Functions Modifying a vector

Write another version of the function **unique(v)** which removes all duplicates from the existing **vector<int>** **v**. Paste the results in the text areas below.

|  |
| --- |
| *Copy and paste a screenshot of your source code here* |

|  |
| --- |
| *Copy and paste a screenshot of the test results* |