**DATA STRUCTURES & ALGORITHMS**

**MACHINE PROBLEM**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Group Name:**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Group Members:** | ***Surname first, alphabetically arranged…***   |  |  | | --- | --- | | **Name** | **Role/Contribution** | |  |  | |  |  | |  |  | |  |  | |
| **Section:** |  |
| **Professor:** |  |
| **Date:** |  |

**INSTRUCTIONS**

* This machine problem is a **GROUP ACTIVITY**.
* You will submit both **the program and documentation**. See details below:
  + **For the Documentation**
    - The deadline of submission will be during your defense schedule.
    - The document must contain the following:
      * A **Title Page** see format below.
      * You must provide the following:
        + **Flowchart** of the Program Logic
        + A **table summarizing all the data structures/ADT used, algorithms, functions, files and classes** with corresponding description.
        + A **table summarizing the division of labor** per team member.
        + **Program Listings / source codes**.
        + **Print Screens of Sample Outputs**. **5 for Valid Inputs and 5 for Invalid Inputs**
      * Use short bond papers and short folders. Use short folder with **clear cover** **in front** and the follow the folder back color below according to your section:
        + N01 – RED
        + IR01 – GREEN
        + IW01 – YELLOW
        + IR02/IW02/N02 – BLUE
      * Follow the format specifications below:
        + Font Size – 10 pt.
        + Font – Calibri
        + Left, Right, Top, Bottom Margin – 1”
        + Paragraph Spacing – Single
* **WARNING: Any GROUP found to submit copied programs (that is the logic used is similar to other groups across all the sections regardless of changing variable names and other identifiers) will automatically get a score of 0 and will not be scheduled for defense. ALL GROUPS (both the group that copied & the team that allowed copying) found to be guilty of this offense will suffer the corresponding consequences.**

**SAMPLE TITLE PAGE**

**CCS0015-Data Structures & Algorithms**

**(font size 26)**

**MACHINE PROBLEM**

**(font size 36)**

**(font size 20)**

**<Section>**

***Submitted By:***

**[Group Name]**

**<Surname, Firstname MI.>**

**<Surname, Firstname MI.>**

**<Surname, Firstname MI.>**

**<Surname, Firstname MI.>**

***Submitted To:***

**<Professor>**

**<Defense Date>**

**FORMATS OF REQUIRED TABLES**

**Table of Functions**

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Scope**  **(Local / Global)** | **Purpose** |
|  |  |  |
|  |  |  |

**Table of Classes**

|  |  |  |
| --- | --- | --- |
| **Class Name** | **Scope**  **(Public /Private)** | **Purpose** |
|  |  |  |
|  |  |  |

**Table of Data Structures/ADT**

|  |  |  |
| --- | --- | --- |
| **Identifier Name** | **Type of Data Structure/**  **ADT Operations** | **Purpose** |
|  |  |  |
|  |  |  |

**Table of Algorithms**

|  |  |  |
| --- | --- | --- |
| **Algorithm Name** | **In what part of the program was it applied** | **Purpose** |
|  |  |  |
|  |  |  |

**Division of Labor**

|  |  |  |
| --- | --- | --- |
| **Function Name** | **Author/Developer** | **Date Completed** |
|  |  |  |
|  |  |  |

**MACHINE PROBLEM**

**PROGRAM DESCRIPTION**

A new video store in your neighborhood is about to open. However, it does not have a program to keep track of its videos and customers. The store managers want someone to write a program for their system so that the video store can operate.

The program will require you to design **2 ADTs** as described below:

|  |  |
| --- | --- |
| **[1] VIDEO ADT** | |
| **Data** | **Operations** |
| Video\_ID (preferably int, auto-generated)  Movie Title  Genre  Production  Number of Copies  Movie Image Filename | [1] Insert a new video  [2] Rent a video; that is, check out a video  [3] Return a video, or check in, a video  [4] Show the details of a particular video  [5] Display all videos in the store  [6] Check whether a particular video is in the store |

|  |  |
| --- | --- |
| **[2] CUSTOMER PARENT ADT** | |
| **Data** | **Operations** |
| Customer\_ID (preferably int, auto-generated)  Name  Address | [1] Add Customer  [2] Show the customer details  [3] Print list of all customers |
| **[3] CUSTOMER-RENT CHILD ADT** | |
| Customer\_ID (  Video\_ID (of all rented videos of a customer) | [1] Rent a Video (Add to the video ids rented)  [2] Return a Video (Remove the video id from the list)  [3] Print list of all videos rented by each customer |

The program will require you to maintain **3 text files** with specifications below:

|  |  |
| --- | --- |
| **Text File** | **Requirements** |
| **[1] VIDEO Text File** | [1] Will store the information about the Videos  [2] Should contain the following by default:  5 Horror Movies  5 Romance Movies  5 Sci-Fi Movies  5 Action Movies  5 Comedy Movies  ***\*\*Must be authentic and true.\*\**** |

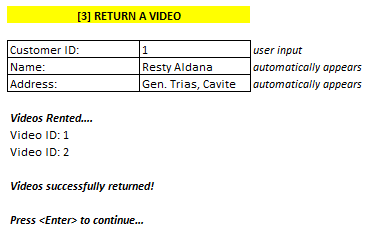
|  |  |
| --- | --- |
| **[2] CUSTOMER Text File** | [1] Will store the basic information about the Customers  [2] Should contain at least 10 customers by default  ***\*\*Must be authentic and true.\*\**** |
| **[3] CUSTOMER-RENT** | [1] Will store all customers that RENT a VIDEO  [2] Will store all the Video\_IDs of all rented videos  [3] Will delete Video\_IDs and Customer\_ID when videos are returned |

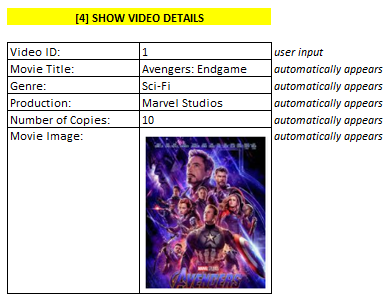
**NOTE: IF YOU FIND THE NEED TO ADD OR MAKE ALTERATIONS TO THE SPECIFICATIONS OF THE ADT AND TEXT FILES ABOVE, PLEASE DO SO. JUST MAKE SURE THAT YOU HAVE AN ACCEPTABLE LOGICAL REASON.**

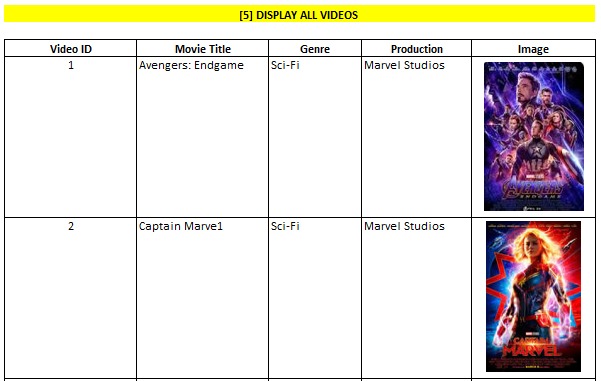
**INSTRUCTIONS:**

* The program will have the following options/menus:
  + **[1] New Video //Programmer1**
  + **[2] Rent a Video //Lead Programmer**
  + **[3] Return a Video //Lead Programmer**
  + **[4] Show Video Details //Programmer2**
  + **[5] Display all Videos //Programmer1**
  + **[6] Check Video Availability //Programmer2**
  + **[7] Customer Maintenance**
    - **[1] Add New Customer //Programmer1**
    - **[2] Show Customer Details //Programmer2**
    - **[3] List of Videos Rented by a Customer //Lead Programmer**
  + **[8] Exit Program //Programmer1**
* Follow the suggested screen dialogs below for each of the option above:

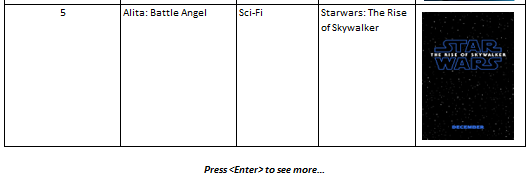


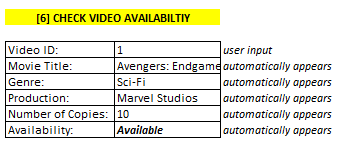


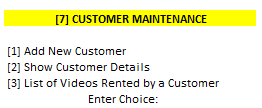


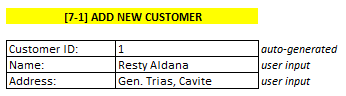


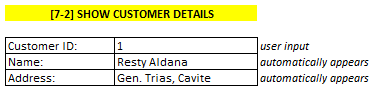


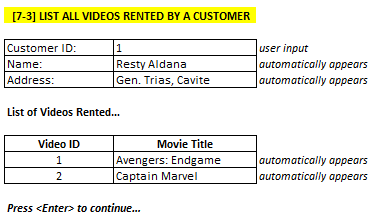








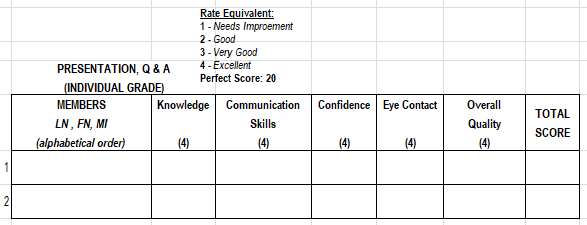




**OTHER REQUIREMENTS:**

* You must store your videos in a **linked list** when you retrieved them from the text file. They must also be stored in a linked list data structure during processing. Saving back to the text file will be done when the user chooses [8] Exit Program.
* You must store you customers in a **queue** when you retrieved them from the text file. They must also be stored in a queue during processing. Saving back to the text file will be done when the user chooses [8] Exit Program.
* Rented videos will be stored in a stack and will be saved in the CUSTOMER-RENT text file when the user chooses [8] Exit Program.
* Use **data structures, files, functions, ADTs, STLs and algorithms** in your program**.** Use **STLs** inside your **ADTs**.
* Use existing and appropriate **sorting** and **searching** algorithms.
* Put **necessary comments** to your program.
* **No global declarations**.
* Provide error messages whenever necessary.

**PRESENTATION RUBRIC:**



**SUBMISSION & DEFENSE SCHEDULE: WEEKS 11-12**

**PROGRAM RUBRIC:**

|  |  |
| --- | --- |
| Department | CCSMA |
| Subject Code | CCS0015 |
| Description | Data Structures & Algorithms |
| Term/Academic Year | 3rd Term SY 2018-2019 |

|  |  |
| --- | --- |
| Topic | Modules 1-10 |
| Project | 1 |
| Project Activity | **Project Presentation** |
| CLO | **1, 2, 3, 4** |

**Note: The following rubrics/metrics will be used to grade students’ output in the project.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Criteria | Exceptional | Acceptable | Amateur | Unsatisfactory |
| Specifications  (40%) | The project works and meets all of the specifications. (40) | The project works and produces the correct results and displays them correctly. It also meets most of the other specifications. (35-39) | The project produces correct results but does not display them correctly. (30-34) | The project is producing incorrect results. (20-29) |
| Design  (15 %) | The design is exceptionally attractive. Project is "user-friendly" with informative and consistent prompts and messages. (15) | The design is fairly attractive. Project is "user-friendly" with informative and consistent prompts and messages. (13-14) | The design is fairly attractive. Project is not "user-friendly" but still provides informative and consistent prompts and messages. (10-12) | The design is unattractive and not user-friendly (8-9) |
| Efficiency (20%) | The code is extremely efficient without sacrificing readability and understanding. (20) | The code is fairly efficient without sacrificing readability and understanding. (17-19) | The code is brute force and unnecessarily long. (14-16) | The code is huge and appears to be patched together. (10-13) |
| Readability  (10 %) | The code is exceptionally well organized and very easy to follow. (10) | The code is fairly easy to read. (8-9) | The code is readable only by someone who knows what it is supposed to be doing. (6-7) | The code is poorly organized and very difficult to read. (4-5) |
| Documentation  (15%) | The team provides written documentation that clarifies their experimental results for that week. Objectives for the past week are clearly stated in relation to project goals, and the report is structured around their accomplishment. (15) | Written documentation is generally complete, but occasional omissions create some lack of clarity. Objectives for the past week are included with no relationship to project goals. (13-14) | Written documentation is incomplete, some lack of clarity. Objectives for the past week are included with no relationship to project goals. (10-12) | There is no supporting written documentation. (8-9) |
| Total: 100% |  |  |  |  |