### SWE20004 Technical Software Development Semester 1 2019

# Extra Credit Report

## Name: Franz Tamani Student ID: 102114822

## Lab class: Thurs / 4:30 / EN303 Tutor: Gavin Chan

**Due Date: Monday 28th May 2019 at 11:59 pm**

**Date Submitted: Tuesday 21th May 2019 at 9:30pm**

## Assignment Title: Extra Credit

## Program Description

The program allows the user to enter/PUSH games into the system using a linked list. It allows the user to enter at the head and the tail of the list. It also allows the user to POP/delete games from the list via the head, the tail of the list, and by a specified name input by the user. It can add customer orders by enqueueing them onto an orders queue. All orders queued are pushed/queued at the back of the queue because it is a first come first served basis, orders can be deleted by dequeuing which adds them to the last sold stack. User can display all games in the system, all orders in the system as well as the last orders that were made in the system.

## Inputs and Outputs

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Data to be stored** | **Sample data** | **Type of data** | **C++ type** | **Input method** | **In / Out** | **Variable name** |
| Menu state | 1, 2, 3 | Integer | int | cin | In | menuOption |
| Game Name | This Game | word | string | cin | In | title |
| Game Genre | 1, 2, 3 | number | enum | cin | In | genre |
| Delete Specified | This Game | word | string | cin | In | name |
| Find Specified | This Game | word | string | cin | In | name |
| Enqueue Order | This Game | word | string | cin | In | name |
| Current Element | Title,  1, Game\* | structure | struct | cout | Out | current |

## Source Code

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  enum Genre { FPS, RPG, PUZZLE};  int GenreAmt = 3; //How many genres there are  struct GameList  {  string title;  Genre gen;  struct GameList\* next;  };  typedef struct GameList Game;  typedef Game\* GamePtr;  void menu()  {  cout << "-------------MENU-------------" << endl;  cout << "1. Insert Game at the Head" << endl;  cout << "2. Insert Game at the Tail" << endl;  cout << "3. Delete Game from the Head" << endl;  cout << "4. Delete Game from the Tail" << endl;  cout << "5. Delete Specific Game" << endl;  cout << "----------Order Menu----------" << endl;  cout << "6. Enqueue Customer Order" << endl;  cout << "7. Dequeue Customer Order" << endl;  cout << "-----------DISPLAY------------" << endl;  cout << "8. Display Games in System" << endl;  cout << "9. Display Orders in System" << endl;  cout << "10. Display Last Sold" << endl;  cout << "11. Exit" << endl;  cout << "Choice: ";  }  GamePtr create\_game(string\* genres)  {  GamePtr newGame = new Game();  unsigned int genre;  cout << "Enter Game Title: ";  cin >> newGame->title;  for (int i = 0; i < GenreAmt; i++)  {  cout << i + 1 << ". " << \*genres << endl;  genres++;  }  cin >> genre;  newGame->gen = static\_cast<Genre>(genre - 1);  return newGame;  }  // PUSH NEW GAME AT HEAD  GamePtr insert\_at\_head(GamePtr &game, string\* genres)  {  GamePtr newGame = new Game();  GamePtr temp = new Game();  newGame = create\_game(genres);  // If there are no games  if (game == NULL)  newGame->next = NULL;  else  {  //Make newGame top of stack  temp->title = game->title;  temp->gen = game->gen;  temp->next = game->next;  newGame->next = temp;  }  return newGame;  }  // PUSH NEW GAME AT TAIL  // Use & to allow changes on the variable  void insert\_at\_tail(GamePtr &game, string\* genres)  {  GamePtr newGame = create\_game(genres);  newGame->next = NULL;  if (game != NULL)  {  GamePtr temp = game;  //Cycle through till the end/tail  while (temp->next != NULL)  {  temp = temp->next;  }  //Make newGame tail of stack  temp->next = newGame;  }  //if there are no games in systems  else  game = newGame;  }  // POP GAME FROM HEAD  // Use & to allow changes on the variable  void delete\_from\_head(GamePtr &gameTop)  {  // Make next game the top of list  if (gameTop != NULL)  {  gameTop = gameTop->next;  }  // No Games  else  {  cout << "There are no games to delete" << endl;  }  }  // POP GAME FROM TAIL  // Use & to allow changes on the variable  void delete\_from\_tail(GamePtr &gameTop)  {  // No Games  if (gameTop == NULL)  {  cout << "There are no games to delete" << endl;  }  else  {  GamePtr current = gameTop;  GamePtr previous = gameTop;  //Cycle through the stack and make last  //element NULL  while (current->next != NULL)  {  previous = current;  current = current->next;  }  previous->next = NULL;  }  }  // Use & to allow changes on the variable  void delete\_specific\_element(GamePtr &gameTop)  {  //No Games  if (gameTop == NULL)  {  cout << "There are no games to delete" << endl;  }  else  {  //Finds game according to specified name and deletes  string name;  bool found = false;  GamePtr current = gameTop;  GamePtr previous = gameTop;  cout << "Enter Game Title: ";  cin >> name;  cin.ignore();  //Checks all elements in list for match  do  {  if (current->title == name)  {  //If game specified is first  //Make top next game  if (current == gameTop)  {  gameTop = gameTop->next;  found = true;  break;  }  //Removes game from list  else  {  previous->next = current->next;  found = true;  break;  }  }  previous = current;  current = current->next;  } while (!found || current != NULL);  //Game not found  if (!found)  cout << name << " Could Not Be Found!" << endl;  }  }  //Find Game for Orders  Game\* findGame(GamePtr gameTop, string name)  {  GamePtr order = NULL;  //If no games  if (gameTop == NULL)  {  cout << "There are no games to order" << endl;  return order;  }  else  {  Game\* current = gameTop;  //Cycle through all elements  do  {  //Game is found, return it  if (current->title == name)  {  order = new Game();  order->title = current->title;  order->gen = current->gen;  order->next = NULL;  return order;  }  current = current->next;  }while (current != NULL);  //No game found  cout << name << "Game Could Not Be Found!" << endl;  return order;  }  }  //QUEUE ORDER AT THE END BECAUSE FIRST COME FIRST SERVED  // Use & to allow changes on the variable ordersTop  void enqueue\_order(GamePtr gameTop, GamePtr &ordersTop)  {  GamePtr order;  string name;  cout << "Title of Game Ordered: ";  cin >> name;  //Find Game With Title  //Check if null/game not found  if (findGame(gameTop, name) == NULL)  return;  //If not null order as game  else  order = findGame(gameTop, name);  //If game could not be found exit function  if (order == NULL)  return;  //If there are no orders  if (ordersTop == NULL)  ordersTop = order;  //QUEUE ORDER AT THE END BECAUSE FIRST COME FIRST SERVED  else  {  GamePtr current = ordersTop;  //Traverse queue all the way to the end  while (current->next != NULL)  {  current = current->next;  }  //Once end of queue reached  current->next = order;  }  }  // Use & to allow changes on the variable  void dequeue\_order(GamePtr &ordersTop, GamePtr &lastSold)  {  //If order queue is empty  if (ordersTop == NULL)  {  cout << "There are no orders" << endl;  }  else  {  Game\* sold = new Game;  sold->title = ordersTop->title;  sold->gen = ordersTop->gen;  sold->next = NULL;  //NO previously sold  if (lastSold == NULL)  lastSold = sold;  //Make last sold first in sold list  else  {  GamePtr temp = new Game;  temp->title = lastSold->title;  temp->gen = lastSold->gen;  temp->next = lastSold->next;  sold->next = temp;  lastSold = sold;  }  // POP Order for Order Queue  ordersTop = ordersTop->next;  }  }  int main()  {  string title;  //Used for printing  string genres[] = { "FPS", "RPG", "PUZZLE"};  GamePtr gameTop = NULL;  GamePtr ordersTop = NULL;  GamePtr lastSold = NULL;  GamePtr current;  int i = 0;  unsigned int menuOption = 0;  while (menuOption != 11)  {  switch (menuOption)  {  case 0:  menu();  cin >> menuOption;  break;  case 1:  //PUSH / Insert game at top  gameTop = insert\_at\_head(gameTop, genres);  menuOption = 0; // Back to Menu  break;  case 2:  //PUSH / Insert game at tail  insert\_at\_tail(gameTop, genres);  menuOption = 0; // Back to Menu  break;  case 3:  //POP/delete game at top  delete\_from\_head(gameTop);  menuOption = 0; // Back to Menu  break;  case 4:  //POP/delete game at tail  delete\_from\_tail(gameTop);  menuOption = 0; // Back to Menu  break;  case 5:  //POP/delete specified game  delete\_specific\_element(gameTop);  menuOption = 0; // Back to Menu  break;  case 6:  //PUSH / QUEUE game order  enqueue\_order(gameTop, ordersTop);  menuOption = 0; // Back to Menu  break;  case 7:  //POP/delete order  dequeue\_order(ordersTop, lastSold);  menuOption = 0; // Back to Menu  break;  case 8:  //Displays all games in system  current = gameTop;  cout << endl << "--------------------------------------------------" << endl;  cout << "GAMES IN THE SYSTEM" << endl;  cout << "--------------------------------------------------" << endl;  while (current != NULL)  {    cout << ++i << ". " << current->title << endl;  current = current->next;  }  cout << "--------------------------------------------------" << endl;  i = 0;  menuOption = 0; // Back to Menu  break;  case 9:  //Displays all games ordered  current = ordersTop;  cout << endl << "--------------------------------------------------" << endl;  cout << "ORDERS IN THE SYSTEM" << endl;  cout << "--------------------------------------------------" << endl;  while (current != NULL)  {  cout << ++i << ". " << current->title << endl;  current = current->next;  }  cout << "--------------------------------------------------" << endl;  i = 0;  menuOption = 0; // Back to Menu  break;  case 10:  //Displays all games sold  current = ordersTop;  cout << endl << "--------------------------------------------------" << endl;  cout << "LAST GAMES SOLD" << endl;  cout << "--------------------------------------------------" << endl;  while (current != NULL)  {  cout << ++i << ". " << current->title << endl;  current = current->next;  }  cout << "--------------------------------------------------" << endl;  i = 0;  menuOption = 0; // Back to Menu  break;  default:  break;  }  }  return 0;  } |

## Test Cases / Screen Shots

|  |
| --- |
| **Push at Head** |

|  |
| --- |
| **Push at Tail** |

|  |
| --- |
| **Delete from Head** |

|  |
| --- |
| **Delete from Tail** |

|  |
| --- |
| **Delete Specific Game** |

|  |
| --- |
| **Enqueue Customer Order** |

|  |
| --- |
| **Dequeue Orders** |

|  |
| --- |
| **Display Last Sold** |