**PASS BY REFERENCE-TO-POINTER AND LINKED LISTS**

| OBJECTIVES  **1. Find the odd and even elements in an integer array**  **2. Create, traverse, add nodes to a linked list**  **3. Get practice implementing classes** |
| --- |

## **Instructions:**

Write code to complete **Problems 1 and 2**. Implement each of the problems as separate programs. To receive credit for your code, you will need to paste your solution into the **Canvas autograder (CodeRunner)**. Your code needs to compile and pass the given test cases in order to receive points. Note that the autograder submissions are divided into separate parts, so that you will be required to test the individual functions prior to testing the entire program.

Please read all the directions ***before*** writing code, as this write-up contains specific requirements for how the code must be written.

## **Problem 1**

## **Pass by Reference-to-Pointer** - Find the odd and even elements in an integer array

**Overview:** In this problem, you will find the odd and even elements in an integer array, and store them in two separate integer arrays passed to the function by reference-to-pointer. We will call this function **findOddAndEvenNumbers**. This function will iterate over an input array to:

1. Find the number of odd elements and even elements in the array.
2. Allocate the required amount of memory to the odd and even arrays respectively.
3. Copy the odd numbers into the odd array
4. Copy the even numbers into the even array.

In case the input array has no even/odd numbers, we will not allocate any memory to the even/odd array, and its pointer will point to nullptr.

**Specifications:**

**1.** Write a function called **findOddAndEvenNumbers**.

**i. INPUT PARAMETERS:** It should take **six** arguments -

* 1. **numbers** is an input array of type int which holds a list of numbers. A pointer to the array is passed to your function.
  2. **length** is the length of the **numbers** array.

**ii. OUTPUT PARAMETERS:** The **findOddAndEvenNumbers** should not return anything.

1. **odd** is an array pointer for an array of type int. A reference to this pointer is passed to the function. **odd** must be updated to point to an array of size equal to the number of odd numbers, and must hold all the odd numbers from the **numbers** array.
2. **even** is an array pointer for an array of type int. A reference to this pointer is passed to the function. **even** must be updated to point to an array of size equal to the number of even numbers, and must hold all the even numbers from the **numbers** array.
3. **numOdd** must be updated to the number of odd numbers in the **numbers** array.
4. **numEven** must be updated to the number of even numbers in the **numbers** array.

**2.** Write a **complete program** to do the following **:**

**i.** Use the function prototype provided below:

| **void findOddAndEvenNumbers(int\* numbers, int\*& odd, int\*& even, int length, int& numOdd, int& numEven);** |
| --- |

**Note**

Notice we don’t return anything, and instead, we use the parameters passed by reference, to update its contents. In C++, we can use parameters passed by reference whenever we want a function to return multiple variables. Here, passing parameters as a reference to a pointer is very important as without it, any updates to the pointers themselves inside the function will not persist inside the main function.

**Problem 2 is on the next page**

## 

## **Problem 2: Linked Lists** Communication Between Persons

**Overview:** In this problem you’re going to model a communication network among persons using a linked list. Each node in the list will represent a person and you need to be able to send a message between nodes from one side of the campus to the other.

**Specifications:**

1. **Building your own communications network**

You will be implementing a class to simulate a linear communication network between persons. There are three files on Moodle containing a code skeleton to get you started. You will have to complete the TODOs in the class implementation in ***SocialMediaNetwork.cpp.***  The driver file ***main.cpp*** is provided to test your code. ***Do not modify the header file or your code won’t work in the Coderunner!***

The linked-list itself will be implemented using the following struct (already included in the header file):

| struct SocialMediaProfile  {  string name; // name of the person  string socialMediaStatus; // status set by the profile  int numberMessages; // no. of messages in the inbox  SocialMediaProfile \*next; // pointer to the next profile  int totalFriends; // number of friends of this profile  }; |
| --- |

### **Class Specifications**

The **SocialMediaNetwork** class definition is provided in the file **SocialMediaNetwork*.hpp*** in Canvas. ***Do not modify this file or your code won’t work on Coderunner!***

Fill in the file **SocialMediaNetwork*.cpp*** according to the following specifications.

1. **SocialMediaProfile\* head;**

* Points to the first node in the linked list

1. **SocialMediaNetwork();**

* Class constructor; set the head pointer to NULL

1. **void addProfileInfo(SocialMediaProfile\* previous, string profileName, int totalFriends, string status); *// Beware of edge cases***

* Insert a new profile with name **profileName and totalFriends and status**  in the linked list after the profile pointed to by **previous**.
* If **previous** is NULL, then add the new profile to the beginning of the list.
* Print the name of the profile you added according to the following format:

| // If you are adding at the beginning use this:  cout << "adding: " << profileName << " (HEAD)" << endl;  // Otherwise use this:  cout << "adding: " << profileName << " (prev: " << previous->name << ")" << endl; |
| --- |

1. **void loadDefaultSetup();**

* Add the following six profiles along with their number of friends and their status, in order, with **addProfileInfo**: "Marshall", "Lily", "Ted", "Robin", "Barney", "Ranjit". Friend numbers are 10, 8, 6, 5, 4, 15 respectively. Statuses are respectively:
  + “There is a great sandwich place on 24th.”
  + “I want to move to the suburbs.”
  + “Working on a building design right now.”
  + “The Canucks won today.”
  + “Suit up.”
  + “NYC is an interesting city.”

1. **SocialMediaProfile\* searchForProfile(string profileName);** 
   * Return a pointer to the node with name **profileName**. If **profileName** cannot be found, return NULL
2. **void transmitNumFriendsInfo(string receiverProfileName);**

* Traverse the linked list from the head to the node with name r**eceiverProfileName**. For each node in this path (including the head), increment the node’s **numberMessages** field. If the list is empty, print "Empty list" and exit the function. If the receiver node is not present, print “Profile not found”.
* As you traverse the list, at each node print the totalFriends and the number of messages received using the following cout: (See the end of this document for example output)

| cout << node->name << " [# messages received: " << node->numberMessages << "] received: " << node->name << " has " << node->totalFriends << " friends." << endl; |
| --- |

**For a complete example refer to the section Main driver file.**

1. **void printNetwork();**

* Print the **name and totalFriends** of each node/profile in the linked list. Below is an example of correct output using the default setup. (Note that you will **cout << “NULL”** at the end of the path)

| == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Barney(4) -> Ranjit(15) -> NULL  === |
| --- |

* If the network is empty then print *"nothing in path"*

1. **Main driver file**

Your program will start by displaying a menu by calling the **displayMenu** function included in main.cpp. The user will select an option from the menu to decide what the program will do, after which, the menu will be displayed again. The specifics of each menu option are described below.

Kindly note ***main.cpp*** is provided. **You do not have to code the driver.** This document is here for reference.

1. **Option 1: Build Network**
   * This option calls the **loadDefaultSetup** function, then calls the **printNetwork** function. You should get the following output:

| adding: Marshall (HEAD)  adding: Lily (prev: Marshall)  adding: Ted (prev: Lily)  adding: Robin (prev: Ted)  adding: Barney (prev: Robin)  adding: Ranjit (prev: Barney)  == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Barney(4) -> Ranjit(15) -> NULL  === |
| --- |

1. **Option 2: Print Network Path**
   * Calls the **printNetwork** function. Output should be in the format below:

| // Output for the default setup  == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Barney(4) -> Ranjit(15) -> NULL  ===  // Output when the linked list is empty  == CURRENT PATH ==  nothing in path  === |
| --- |

1. **Option 3: Broadcast Profile Info**
   * Prompt the user for one input: The name of a profile to send a message to. Pass the profile name to the **transmitNumFriendsInfo** function. *Don’t forget to add a newline after the receiver is collected, and before the output is printed. This is done for better readability.* Each node should create its message with its name and totalFriends:
   * **“<profileName> + “ has “ + < totalFriends> + “ friends.”**

* Update the **numberMessages** for nodes accordingly.
* For example, the following should be the output if the linked-list contains the default setup from option (1) and receiver is Barney:

| Example 1:  Enter name of the recipient to receive the message:  Robin  Marshall [# messages received: 1] received: Marshall has 10 friends.  Lily [# messages received: 1] received: Lily has 8 friends.  Ted [# messages received: 1] received: Ted has 6 friends.  Robin [# messages received: 1] received: Robin has 5 friends. |
| --- |

* If the user then decides to call the function again with “Ranjit” as receiver :

| Example 2:  Enter name of the recipient to receive the message:  Ranjit  Marshall [# messages received: 2] received: Marshall has 10 friends.  Lily [# messages received: 2] received: Lily has 8 friends.  Ted [# messages received: 2] received: Ted has 6 friends.  Robin [# messages received: 2] received: Robin has 5 friends.  Barney [# messages received: 1] received: Barney has 4 friends.  Ranjit [# messages received: 1] received: Ranjit has 15 friends. |
| --- |

* If the user then decides to broadcast room info to “Kevin”, the output when the profile is not present will be:

| Example 3:  Enter name of the recipient to receive the message:  Kevin  Profile not found |
| --- |

1. **Option 4: Add Profile**
   * Prompt the user for four inputs: the name of a new profile to add to the network, **newProfile**, the number of friends the new profile has, the social media status of the new profile, and the name of a profile already in the network, **previous**, which will precede the new profile. Use the member functions **searchForProfile** and **addProfileInfo** to insert **newProfile** into the linked-list right after **previous**.

* If the user wants to add the new profile to the head of the network then they should enter “First” instead of a previous profile name.
* If the user enters an invalid previous profile (not present in the linked list), then you need to prompt the user with the following error message and collect input again until they enter a valid previous profile name or “First”:

| cout << "INVALID(previous profile name)...Please enter a VALID profile name!" << endl; |
| --- |

* Once a valid previous profile name is passed and the new profile is added, call the function **printPath** to demonstrate the new linked-list.
* The Names of the profiles are case-sensitive.
  + For example, the following should be the output if the linked-list contains the default setup from option (1) and the user wants to add Kevin after Robin

| Enter a new profile name:  Kevin  Enter number of friends:  8  Enter the social media status:  I love soccer!  Enter the previous profile name (or First):  Robin  adding: Kevin (prev: Robin)  == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Kevin(8) -> Barney(4) -> Ranjit(15) -> NULL  === |
| --- |

1. **Option 5: Quit**

* Print the following message:

| cout << "Quitting..." << endl; |
| --- |

* Finally, print the following before exiting the program:

| cout << "Goodbye!" << endl; |
| --- |

1. **Example run**

| Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 2  == CURRENT PATH ==  nothing in path  ===  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 3  Enter name of the recipient to receive the message:  Robin  Empty list  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 1  adding: Marshall (HEAD)  adding: Lily (prev: Marshall)  adding: Ted (prev: Lily)  adding: Robin (prev: Ted)  adding: Barney (prev: Robin)  adding: Ranjit (prev: Barney)  == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Barney(4) -> Ranjit(15) -> NULL  ===  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 2  == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Barney(4) -> Ranjit(15) -> NULL  ===  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 3  Enter name of the recipient to receive the message:  Kevin  Profile not found  Select a numerical option:  +=====Main Menu=========+  Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 4  Enter a new profile name:  Kevin  Enter number friends:  8  Enter the social media status:  I love soccer!  Enter the previous profile name (or First):  Robin  adding: Kevin (prev: Robin)  == CURRENT PATH ==  Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Kevin(8) -> Barney(4) -> Ranjit(15) -> NULL  ===  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 3  Enter name of the recipient to receive the message:  Robin  Marshall [# messages received: 1] received: Marshall has 10 friends.  Lily [# messages received: 1] received: Lily has 8 friends.  Ted [# messages received: 1] received: Ted has 6 friends.  Robin [# messages received: 1] received: Robin has 5 friends.  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 3  Enter name of the recipient to receive the message:  Ted  Marshall [# messages received: 2] received: Marshall has 10 friends.  Lily [# messages received: 2] received: Lily has 8 friends.  Ted [# messages received: 2] received: Ted has 6 friends.  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 4  Enter a new profile name:  Stella  Enter number friends:  10  Enter the social media status:  Hello!  Enter the previous profile name (or First):  First  adding: Stella (HEAD)  == CURRENT PATH ==  Stella(10) -> Marshall(10) -> Lily(8) -> Ted(6) -> Robin(5) -> Kevin(8) -> Barney(4) -> Ranjit(15) -> NULL  ===  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 4  Enter a new profile name:  Victoria  Enter number friends:  2  Enter the social media status:  CR7 is back to Man Utd.  Enter the previous profile name (or First):  Chris  INVALID(previous profile name)...Please enter a VALID profile name!  Ted  adding: Victoria (prev: Ted)  == CURRENT PATH ==  Stella(10) -> Marshall(10) -> Lily(8) -> Ted(6) -> Victoria(2) -> Robin(5) -> Kevin(8) -> Barney(4) -> Ranjit(15) -> NULL  ===  Select a numerical option:  +=====Main Menu=========+  1. Build Network  2. Print Network Path  3. Broadcast Profile Info  4. Add Profile  5. Quit  +-----------------------+  #> 5  Quitting...  Goodbye! |
| --- |

1. **Appendix:**

* **SocialMediaNetwork::addProfileInfo()** 
  + cout << "adding: " << profileName << " (HEAD)" << endl;
  + cout << "adding: " << profileName << " (prev: " << previous->name << ")" << endl;
* **SocialMediaNetwork::transmitFriendsInfo(string receiverProfileName)**
  + cout << "Empty list" << endl;
  + cout << "Profile not found" << endl;
  + cout << node->name << " [# messages received: " << node->numberMessages << "] received: " << node->name << " has " << node->totalFriends << " friends." << endl;
* **SocialMediaNetwork::printNetwork()**
  + cout << "== CURRENT PATH ==" << endl;
  + cout << "nothing in path" << endl;
  + cout << ptr->name << "(" << ptr->totalFriends << ")" <<" -> ";
  + cout << "NULL" << endl;
  + cout << "===" << endl;
* **main()**
  + cout << "Enter name of the profile to receive the message: "<< endl;
  + cout << endl;
  + cout << "Enter a new profile name: " << endl;
  + cout << "Enter total friends: " << endl;
  + cout << "Enter the previous profile name (or First): " << endl;
  + cout << "INVALID(previous profile name)...Please enter a VALID profile name!" << endl;
  + cout << "Quitting..." << endl;
  + cout << "Invalid Input" << endl;
  + cout << "Goodbye!" << endl;
* **displayMenu()**
  + cout << endl;
  + cout << "Select a numerical option:" << endl;
  + cout << "+=====Main Menu=========+" << endl;
  + cout << " 1. Build Network " << endl;
  + cout << " 2. Print Network Path " << endl;
  + cout << " 3. Broadcast Profile Info " << endl;
  + cout << " 4. Add Profile " << endl;
  + cout << " 5. Quit " << endl;
  + cout << "+-----------------------+" << endl;
  + cout << "#> ";