CSCI-1200 Data Structures - Spring 2024
Test 1 – Friday, Feburary 1st 6pm-7:50pm

- This exam has 4 problems worth a total of 100 points.
- This packet contains 10 pages numbered 1-10. Please count the pages of your exam and raise your hand if you are missing a page.
- The packet contains 1 blank page, which is page 10. If you use a blank page to solve a problem, make a note in the original box and clearly label which problem you are solving on the blank page.
- **DO NOT REMOVE THE STAPLE OR SEPARATE THE PAGES OF YOUR EXAM. DOING SO WILL RESULT IN A -10 POINT PENALTY!**
- You may have pencils, eraser, pen, tissues, water, and your RPI ID card on your desk. Place everything else on the floor under your chair. Electronic equipment, including computers, cell phones, calculators, music players, smart watches, cameras, etc. is not permitted and must be turned "off" (not just vibrate).
- Raise your hand if you need to ask a proctor something that is NOT related to one of the questions on the test.
- Please state clearly any assumptions that you made in interpreting a question. Unless otherwise stated you may use any technique that we have discussed in lecture, lab, or on the homework.
- Please write neatly. If we can't read your solution, we can't give you full credit for your work.
- You do not need to write `#include` statements for STL libraries. Writing `std::` is optional. The keyword `auto` is not allowed.

Problem 1: std::string. [/20]

Write a function called reverseString() which reverses an std::string object. Do not call the std::reverse() function.

Your code will be tested like this:

```
int main() {
    // Example usage
    std::string original = "Why not change the world?";
    std::string reversed = reverseString(original);

    // Print the results
    std::cout << "Original: " << original << std::endl;
    std::cout << "Reversed: " << reversed << std::endl;

    return 0;
}
```

And this program will output:

```
Original: Why not change the world?
Reversed: ?dlrow eht egnahc ton yhW
```

Solution:

```
std::string reverseString(const std::string& input) {
    // Create an empty string to store the reversed result
    std::string reversed;

    // Iterate through the input string in reverse order
    for (int i = input.length() - 1; i >= 0; --i) {
        // Append each character to the reversed string
        reversed += input[i];
    }

    return reversed;
}
```

Problem 2: Big O Notation. [/20]

What's the runtime complexity of the following functions, in terms of n. You can assume the size of the array arr in Function 1, 2, 3, and 4 is n.

Function 1:

```
int findMax(const std::vector<int>& arr) {
    int max = arr[0];
    for (int i = 1; i < arr.size(); ++i) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}
```

Function 2:

```
int getFirstElement(const std::vector<int>& arr) {
    if (!arr.empty()) {
        return arr[0];
    }
    return -1; // Default value if array is empty
}
```

Function 3:

```
void sort(std::vector<int>& arr) {
    int n = arr.size();
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (arr[j] > arr[j + 1]) {
// this swap function swaps the value between arr[j] and arr[j+1].
std::swap(arr[j], arr[j + 1]);
            }
        }
    }
}
```

Function 4:

```
#include <vector>

int sumElements(const std::vector<int>& arr) {
    int sum = 0;
    for (unsigned int i = 0; i < arr.size(); ++i) {
        sum += arr[i];
    }
    return sum;
}
```

Function 5:

```
int powerOfTwo(int n) {
    if (n == 0) {
        return 1;
    } else {
        return 2 * powerOfTwo(n - 1);
    }
}
```

Solution:
Function 1: O(n)
Function 2: O(1)
Function 3: O(n^2)
Function 4: O(n)
Function 5: O(n)

Problem 3: C++ Class. [/35]

In this problem you will implement Yelp. Yelp is a popular online platform and mobile application that allows users to discover and review local businesses, particularly restaurants. It provides a platform for users to share their experiences and opinions about various businesses, including restaurants, bars, cafes, and other services. To implement Yelp, you will implement a class called Yelp.

To assist you implement Yelp, first, three helper classes are provided for you:

```
// define a User class
class User {
private:
    std::string username;

public:
    // default constructor
    User(){
            username = "";
    }
    // other constructor
    User(const std::string& name) {
            username = name;
    }
    std::string getUsername(){
            return username;
    }
};

// define a Restaurant class
class Restaurant {
private:
    std::string name;

public:
    // default constructor
    Restaurant(){
            name = "";
    }
    // other constructor
    Restaurant(const std::string& n) {
            name = n;
    }
    std::string getName(){
            return name;
    }
};

// define a Review class
class Review {
private:
    User reviewer;
    Restaurant restaurant;
    std::string text;
    // Add other review-related attributes as needed

public:
    Review(const User& u, const Restaurant& r, const std::string& t) {
            reviewer = u;
            restaurant = r;
            text = t;
    }
    User getReviewer(){
            return reviewer;
    }
    Restaurant getRestaurant(){
            return restaurant;
    }
    std::string getText(){
            return text;
    }
};
```

Second, we have provided the program for testing your Yelp class.

```
int main() {
    Yelp yelp;

    // Add users, restaurants, and reviews for testing
    yelp.addUser("User1");
    yelp.addUser("User2");

    yelp.addRestaurant("Restaurant1");
    yelp.addRestaurant("Restaurant2");

    yelp.addReview("User1", "Restaurant1", "Great food!");
    yelp.addReview("User2", "Restaurant1", "Nice ambiance.");

    yelp.addReview("User1", "Restaurant2", "Average experience.");
    yelp.addReview("User2", "Restaurant2", "Excellent service!");

    // Display reviews for a specific restaurant
    yelp.displayReviews("Restaurant1");
    yelp.displayReviews("Restaurant2");

    return 0;
}
```

Your job now is to write the Yelp class so that the above test program will produce the following output:

```
Reviews for Restaurant1:
User1: Great food!
User2: Nice ambiance.
Reviews for Restaurant2:
User1: Average experience.
User2: Excellent service!
```

Note that you are not allowed to use any STL containers which we have not covered in this class, such as std::list, std::map, std::set; but feel free to user std::string and std::vector.

3.1 Yelp Declaration (Yelp.h) [/ 15] Start by writing the header file for the Yelp class.

Solution:

```
// Yelp class to manage users, restaurants, and reviews
class Yelp {
private:
    std::vector<User> users;
    std::vector<Restaurant> restaurants;
    std::vector<Review> reviews;

public:
    // User management
    void addUser(const std::string& username);

    // Restaurant management
    void addRestaurant(const std::string& name);

    // Review management
    void addReview(const std::string& username, const std::string& restaurantName,
                   const std::string& reviewText);

    // Display reviews for a specific restaurant
    void displayReviews(const std::string& restaurantName);

};
```

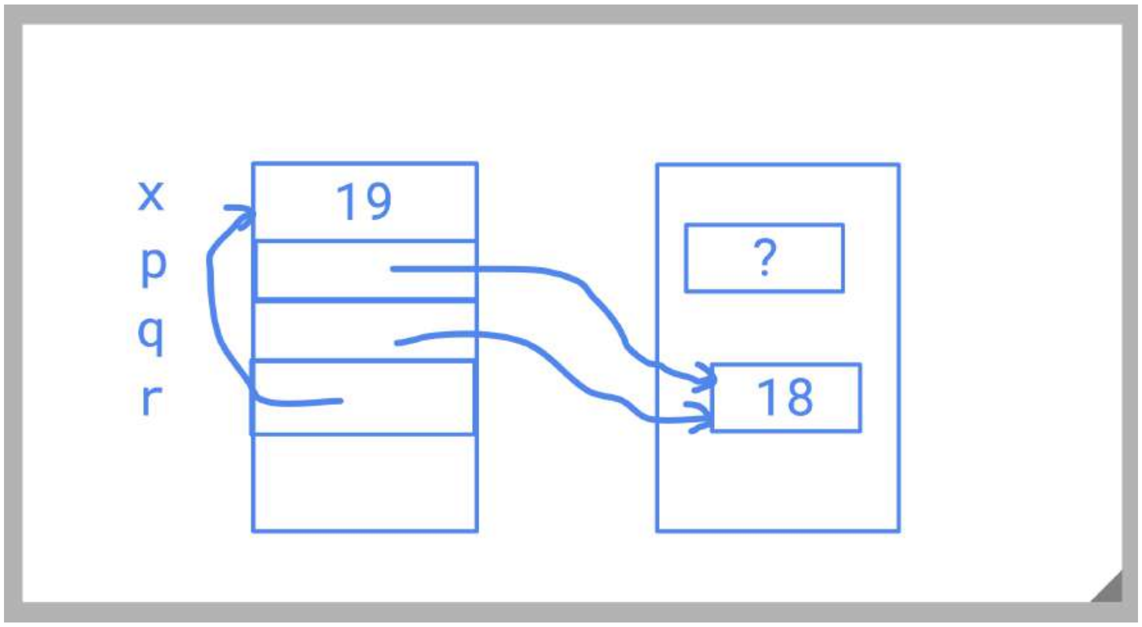
3.2 Yelp Implementation (Yelp.cpp) [/ 20] Next write the implementation of the Yelp class.

```
// User management
void Yelp::addUser(const std::string& username) {
    users.push_back(username);
}

// Restaurant management
void Yelp::addRestaurant(const std::string& name) {
    restaurants.push_back(name);
}

// Review management
void Yelp::addReview(const std::string& username, const std::string& restaurantName,
                     const std::string& reviewText) {
    User user(username);
    Restaurant restaurant(restaurantName);
    Review review(user, restaurant, reviewText);
    reviews.push_back(review);
}

// Display reviews for a specific restaurant
void Yelp::displayReviews(const std::string& restaurantName) {
    std::cout << "Reviews for " << restaurantName << ":\n";
    int size = reviews.size();
    for (int i=0; i<size; i++) {
        if ((reviews[i].getRestaurant()).getName() == restaurantName) {
            std::cout << (reviews[i].getReviewer()).getUsername() << ": "
                      << reviews[i].getText() << "\n";
        }
    }
}
```

Problem 4: Pointers and Dynamic Memory [/25]

part 1: Given the following code snippet, following the conventions used in Data Structures lecture for memory diagramming, draw a diagram of the stack and the heap that result from executing the code snippet. Use a '?' to represent uninitialized values. [/16]

```
int x = 5;
int *p = new int;
int *q = new int;
int *r = &x;
p = q;
*q = 17;
*p = 18;
*r = 19;
std::cout << "x is " << x << std::endl;
std::cout << "*p is " << *p << std::endl;
std::cout << "*q is " << *q << std::endl;
```



part 2: The above code snippet will print 3 messages to STDOUT, please complete these 3 messages: [/9]
Solution:

```
x is 19

*p is 18

*q is 18
```