CSC 212: Data Structures and Abstractions

Spring 2018

University of Rhode Island

Weekly Problem Set #2

Due Wednesday 2/7 before lab. Please turn in neat, and organized, answers hand-written on standard-sized paper **without any fringe**. The only library you're allowed to use in your answers is `iostream`, though you can test with whatever you'd like. The goal is to understand the low-level functionality behind what the STL offers.

1. Draw the array represented by `int arr[5];` use null to denote uninitialized memory.

2. Now redraw the array after this code executes:

```
*arr = 1;
*(arr+2) = 5;
```

3. What is the output of the following code? If it breaks at any point, indicate what went wrong.

```
#include <iostream>

int mystery(int x, int* y) {
    x = x + 10;
    *y = x * 2;
    return x;
}

int* mystery2() {
    int x = 50;
    return &x;
}

int main() {
    int x = 2, y = 3;
    x = mystery(x, &y);
    std::cout << "(x, y): (" << x << ", " << y << ")" << std::endl;
    int* z = mystery2();
    std::cout << "z: " << *z << std::endl;
}
```

4. Define a void function that takes a pointer to an integer variable as a parameter, and increments its value by 10. (Hint: void functions return type is void)

5. Define a function that sets an arbitrary value of an array. The function should have the following prototype: `void set_value(int* arr, const int n, const int idx, const int val);` It should prevent users from inserting values beyond the end of the array.

6. Define a function that creates a new integer array on the heap and returns a pointer to that array. The function should take the length of the array to generate as a parameter. (Hint: this requires the `new` operator!)

7. Define a second function that receives a pointer to a previously allocated array and frees the memory. `void free_mem(int* arr);`

The following exercises are considered *optional*. You don't need to report answers on these.

1. Provide a detailed explanation as to what happened in question 4, can you suggest any ways to fix the issue present in the code?

2. The `std::vector` class uses arrays internally, yet a `std::vector` is capable of dynamically resizing itself. Can you think of how you would write the method `expand`? *This is a challenging question, which is why it is optional, but this type of thinking is highly valuable.*