**C++ Interlude 2 Pointers, Polymorphism & Memory Allocation - Answers**

Top of Form

Bottom of Form

**Content**

**Answers**

**1. A disadvantage of using a fixed array is that you can run out of room for elements. If you use a dynamically allocated array (the textbook calls them resizable arrays), you have the option of re-allocating the array to a larger size.**

**To show that you know how to use a resizable array, convert the following code to use a resizable array. Include code to free the memory used by the array at the end of your code.**

**const int SIZE = 5;  
    int list[SIZE];  
      
    for (int i=0; i<SIZE; i++)  
        list[i] = 2 \* i;  
      
    for (int i=0; i<SIZE; i++)  
        cout << list[i] << endl;**

To use a resizable array, the code is:

    const int SIZE = 5;  
    int \*list = new int[SIZE];     // allocate array  
      
    for (int i=0; i<SIZE; i++)  
        list[i] = 2 \* i;  
      
    for (int i=0; i<SIZE; i++)  
        cout << list[i] << endl;  
      
    delete [] list;                         // de-allocate array

**2. The textbook includes this definition for a PlainBox class template:**

**// Declaration for the class template PlainBox**

**template ItemType>   
class PlainBox  
{  
private:  
   ItemType item;  
     
public:  
   // Default constructor  
   PlainBox();  
     
   // Parameterized constructor  
   PlainBox(const ItemType& theItem);  
     
   // Mutator method that can change the value of the data field  
   void setItem(const ItemType& theItem);  
     
   // Accessor method to get the value of the data field  
   ItemType getItem() const;  
}; // end PlainBox**

**Write C++ code to:**

* + **Declare a default PlainBox object that can hold a string.**
  + **Call the setItem method to store the string "wagon" in your PlainBox object.**

   PlainBox<string> obj;  
   obj.setItem("wagon");

**3. Using the PlainBox class template from question #2, write C++ code to:**

* + **Declare for a pointer that can hold the address of a PlainBox object that holds a string.**
  + **Dynamically allocate a PlainBox object and put its address in your pointer.**
  + **Call the setItem method to store the string "wagon" in your PlainBox object.**

   PlainBox<string> \*ptr;  
   ptr = new PlainBox<string>();  
   ptr->setItem("wagon");