**Question 1:**

**Answer**

Because the variables on both the side of the assignment operator should be independent after assignment operation. If the variable is a pointer or an array, the assigned variable and the new variable will point to the same pointer or array. Our goal is to have independent/separate arrays or pointer/variables after assignment, hence we use copy constructor.

**Question 2:**

**Answer**

1. Some of the member variables of the class are pointers
2. Member functions allocate and release dynamic memory
3. The automatic value semantics of the class is overridden
4. The class has a destructor to return all dynamic memory to the heap

**Question 3:**

**Answer**

Constructor:

bag::bag(const bag& source);

Destructor:

~bag( );

**Question 4:**

**Answer**

    void bag::operator =(const bag& source)  
    // Library facilities used: algorithm  
    {  
  value\_type \*new\_data;

  // Check for possible self-assignment:  
  if (this == &source)  
    return;

  // If needed, allocate an array with a different size:  
  if (capacity != source.capacity)  
  {   
   new\_data = new value\_type[source.capacity];  
   delete [ ] data;  
   data = new\_data;  
   capacity = source.capacity;  
 }

  // Copy the data from the source array:  
  used = source.used;  
  copy(source.data, source.data + used, data);  
    }