a) Output:
B text
A default

B obj1("text");
First, since B(char *s) : A(s) is called and class B inherits class A so A(char *s) will be executed before B(char *s)

A *obj2 = new B(obj1);
With A *obj2, A() is called first before the pointer is pointed to the address of obj1 as B(const B &b) is called with new B(obj1). Since B inherits class A and there's no specify in B(const B &b), A() is executed before the body of B(const B &b).

foo(&obj1, *obj2)
In the argument, we pass &obj1 from class B and *obj2 from class A to A* obj1 and A obj2 respectively. In the body, obj1->display() calls the display function in class A. In its body, prepare() calls out the prepare() function of class B due to the virtual prepare function in class A. Now obj2.display() calls display() function in the class A and the prepare() function in display() won't be overridden since obj2 is not a pointer.

b) Since the constructor is just simply pointing to the address of char array, it may be dangerous and could make the program crash if there are changes in that address. We should better change the method of constructor: make a copy version.

```
A(char* s){
      if (s){
         m_s= strdup(s);
      }
   }
~A(){
   if (m_s)
   {
      delete[] m_s;
      m_s=nullptr;
   }
}
```

c)
Answer:

```
istream& operator>>(istream &in,A& x){
```

```
    char *s= new char[100];
    if (in >> s)
    {
        x = A(s);
    }
    delete[] s;
    s=nullptr;
    return in;
}
```