## SECTION A: Short Questions

1. What's inheritance and why we use inheritance (e.g. advantage)?

2. What's function overloading? What's function overriding?

3. Concepts of a) polymorphism b)static binding c) dynamic binding.

4. Concepts of pure virtual function and abstract class.

5. What is template and why we use template (e.g. advantage) ?

6. **Virtual DeConstructor**:

Can `delete` *px* free `p` and `pp` in main()? Give a solution to this problem.

```
class X{
private:
      char *p;
public:
      X(int sc){p=new char[sz];}
      ~X() { delete[] p;}
}


int main()
{
X *px = new Y(10,12);
// …
delete px ;
}
```

```
class Y: public X{
private:
      char *pp;
public:
       Y(int sz1,int sz2):X(sz1)
       {
            pp = new char [sz2];
       }
       ~Y(){ delete [ ]pp; }
}
```

7. **Mutiple Inheritance:**

Briefly discuss diamond problem with the following code and give a solution by modifying the code.

```
class Animal { /* ... */ }; // base class
{
      int weight;
public:
      int getWeight() { return weight;};
};

class Tiger : public Animal { /* ... */ };
class Lion : public Animal { /* ... */ }
class Liger : public Tiger, public Lion { /* ... */ };

int main( )
{
      Liger lg ;
      int weight = lg.getWeight();
}
```

## Section B: Programming Questions:

The following code implements a dynamic array, which is similar as ***vector*** container in STL. Please finish the TODO part here.

```
template <class T>
class MyArray
{
      int len;
      T *data;
   public:
         MyArray(int len=0,T* data=NULL);
         ~MyArray();
         T& operator [](int index);
         void push(T d); //add new element
};

template <class T> T& MyArray<T>::operator [](int index)
{
      //TODO check the index range and return the element

}


template <class T> void MyArray<T>::push(T d)
{
             T *pdata = data;
            data = new T[len + 1];
            if(pdata != NULL)
            {
                  for(int i = 0 ; i < len ; i++)
                  {
                           data[i] = pdata[i];
                  }
                  delete[] pdata;
            }
               data[len] = d;
                 len++;
}
```