Examen de Lenguajes de Programación II (E28, F30)

Convocatoria de Febrero de 1997

Consideraciones previas:
* Las preguntas deben ser resueltas en hojas separadas.
* La duración del examen es de tres horas.
* No se puede abandonar el aula sin entregar el examen.

1. Dada la siguiente clase:

```
class Usuario
{
 char *Nombre;
 char *Codigo;
public:
 Usuario ( char *Name, char *Cod );
 Rvalues ( char *Name, char *Cod ); //Devuelve valores
 Wvalues ( char *Name, char *Cod ); //Asigna valores
 ~Usuario ( ) { delete [] Nombre; delete [] Codigo; }
};
```

y suponiendo que el código de las anteriores funciones ya está escrito, añadir las funciones necesarias para que el siguiente fragmento de programa funcione correctamente:

```
Usuario funcion ( Usuario U1 )
{
 Usuario U2;
 char c;

 cout << "El usuario actual es: " << U1 << endl;
 cout << "¿Desea cambiar de usuario?" << endl;
 cin >> c;
 if ( c=='s' || c=='S' )
 {
   cout << "Introduzca el nuevo usuario: ";
   cin >> U2;
   if ( U1 == U2 )
       cout << "Sigue siendo el mismo usuario" << endl;
   else
       U1 = U2;
 }
 return U1;
}
```

Nota: No es necesario generar o controlar errores de entrada/salida.

2. La clase **`Vector`** que se muestra a continuación permite crear vectores de enteros de talla arbitraria *N* cuyos elementos llevan índices en el rango [0..*N*-1]. Teniendo en cuenta los principios de reutilización de código, implementar la clase VectorConRango de forma que se permita crear vectores de enteros con cualquier rango de índices y con la misma funcionalidad que en la clase Vector.

```
class Vector {
 int *p;
 int talla;
public:
 Vector();                 //crea un vector vacío
 Vector(Vector& v);        //constructor copia
 Vector(const int a[], int n);
                           //convierte un array normal a de
                           //de talla n en Vector
 ~Vector();                //destructor
 int max_ind();            //devuelve el índice máximo
 int& operator[](int i);
                           //operador de indexación
 Vector& operator=(Vector& v);
                           //operador de asignación
};
```

3. Reescribir la declaración de la clase Vector del ejercicio anterior con el objetivo de poder declarar vectores con elementos de cualquier tipo e implementar los constructores y el operador de asignación.

Según la implementación propuesta, el siguiente fragmento de código ¿planteará algún problema? En caso afirmativo, ¿cómo puede solucionarse?

```
char *a[3];

a[0]=new char[10]; strcpy(a[0],"string 0");
a[1]=new char[10]; strcpy(a[1],"string 1");
a[2]=new char[10]; strcpy(a[2],"string 2");

Vector<char*> miVector(a,3);
```

4. Responder brevemente a las siguientes preguntas:

1. ¿Qué ventajas reporta la utilización de new y delete frente a malloc() y free()?
2. ¿Qué es una función virtual pura?
3. Con una herencia del tipo protected, ¿cómo son los niveles de acceso de los miembros heredados en la clase derivada?
4. ¿Puede definirse el operador ** para definir la operación *elevar a una potencia*?

Puntuación de las preguntas:

1. 2,5 puntos
2. 2,5 “
3. 3 “
4. 2 “