TEMA 8: GESTIÓN DINÁMICA DE MEMORIA

CUESTIONES

1. Siendo p un puntero a entero, señalar el bloque de sentencias que no presenta ningún error de compilación, ejecución o lógico:

```
a) p = new int; p = NULL;
b) p = NULL; *p = *p + 1;
c) p = new int; *p = 0;
d) delete p; *p = 0;
```

2. Sea la siguiente declaración de tipos:

```
typedef float * Ptr;
struct Reg {
        int c1,c2;
        Ptr c3[100];
};
typedef Reg V[10][20];
```

Si x es una variable de tipo V, indicar el tipo resultante de las siguientes expresiones de acceso o si son incorrectas y por qué:

```
x[1].c3                 _____________________
* x[1][15].c3[5]        _____________________
x.c3[1][1]              _____________________
*(x[5][10].c3[0])       _____________________
x[5][19].c3[50]         _____________________
x[2][12].c2             _____________________
```

3. Dado el siguiente código:

```
int main()
{  int *p,*q,*r;

   p = new int;
   *p = 6;
   r = new int;
   *r = *p;
   q = p;
   *q = 1;
   cout << *p << *q << *r;
}
```

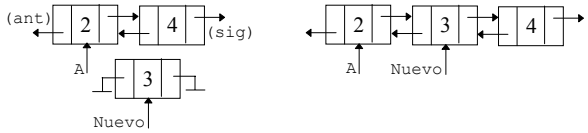
¿Qué valores visualizará su ejecución? _____

---

4. Sea la siguiente estructura de datos:

```
struct Casilla {
       int Contenido;
       Casilla *Ant,*Sig;
};
```

y las siguientes situaciones:



La secuencia correcta de pasos para pasar de la situación inicial a la final es:

```
a) Nuevo->Sig = A->Sig;
   Nuevo->Ant = A;
   A->Sig = Nuevo;
   A->Sig->Ant = Nuevo;

b) A->Sig = Nuevo;
   A->Sig->Ant = Nuevo;
   Nuevo->Sig = A->Sig;
   Nuevo->Ant = A;

c) A->Sig->Ant = Nuevo;
   Nuevo->Sig = A->Sig;
   Nuevo->Ant = A;
   A->Sig = Nuevo;
```

d) Todas las anteriores son falsas.

5. ¿Qué sucederá en el siguiente fragmento de código si p y q son de tipo puntero a entero (`int*`)?

```
p = new int;
*p = 1;
q = p;
delete p;
/*Otras sentencias*/
cout << *q;
```

a) Escribirá en pantalla un 1.

b) Escribirá en pantalla la dirección de memoria a la que apunta `q`.

c) Error lógico, pues no se sabrá el valor de `*q` en la última sentencia.

6. Sea la siguiente declaración de tipos:

```
typedef float * Puntero;
struct Regist {
       String Nombre;
       Puntero Datos;
};
typedef Regist Muestra[20];
```

Si V es una variable de tipo Muestra en donde cada campo *datos* apunta a un vector dinámico reservado con `new float[5]`, indicar qué accesos son correctos, junto con el tipo de la componente accedida:

Acceso	Correcto	Tipo
`*((*V)[10].Datos)`	☐	_____
`V[0].Nombre`	☐	_____
`*V`	☐	_____
`V[5].Datos[3]`	☐	_____
`*V[19].Datos`	☐	_____
`V[5].Nombre[1]`	☐	_____

7. ¿Qué ventaja supone el uso de la secuencia de sentencias <p=new tipo; + cualquier conjunto de instrucciones que usen p; +delete p>; frente a las sentencias <p = new tipo; + cualquier conjunto de instrucciones que usen p; + p=NULL;>?

a) Ninguna; son equivalentes.

b) Ninguna. Presenta inconvenientes por no permitir el uso posterior de *p.

c) delete p libera la memoria ocupada por *p y p=NULL no.

d) delete p libera la memoria ocupada por p y p=NULL no.

8. La sentencia `p = &x`

a) Asigna a la variable puntero *p* la dirección de memoria de la variable *x*.

b) Asigna a **p* el valor de *x*.

c) Asigna el contenido de *x* a *p*.

d) Nada de lo anterior es cierto.