PROGRAMACION AVANZADA - Taller2-1

1) ¿Por qué las siguientes instrucciones C/C++ levantarían un error en ejecución?:

```
int *p = NULL;
*p = 100;
```

2) Establezca la instrucción que pondría al apuntador '*p' apuntar a la variable 'a':

```
int a = 100;
int *p = NULL;
```

3) Realice la respectiva representación gráfica (*como se hizo en la sesión de clase*) de la siguiente porción de código C/C++:

```
13      int x = 100;
14      int y = 200;
15      int *p1 = NULL;
16      int *p2 = NULL;
17      int **p3 = NULL;
18
19      p1 = &x;
20      p2 = &y;
21      p3 = &p2;
```

4) Para el código del Ítem (**3**), realice la instrucción que corresponda, tal que cambie el valor de 'y', desde '**p3', asignándole ahora a 'y' el valor apuntado por '*p1'.

5) Para el código del Ítem (**3**), realice la instrucción que corresponda, tal que '**p3' ahora apunte al valor de 'x'.

6) Para el código del Ítem (**3**), es válido pretender que '**p3' apunte **DIRECTAMENTE** a 'x' o a 'y', ¿por qué?

7) Para el código del Ítem (**3**), implemente las sgtes instrucciones:

Cree un apuntador '*p4'
Que '*p4' apunte también a 'x'
Cree un apuntador '**p5'
Que '**p5' apunte también a '*p4'

Modifique el valor (el dato) apuntado indirectamente de '**p3' con el de '**p5'
Muestre el valor (el dato) apuntado indirectamente de '**p3' y el de '**p5'

8) El siguiente código C/C++ por medio del apuntador '*p' mostrara los valores del arreglo 'arr', 1° la posición 1, luego el valor de la posición 0:

```
13      int arr[] = { 10, 20 };
14      int *p = NULL;
15
16      p = arr;
17
18      cout<<*(p+1)<<endl;
19      cout<<*p<<endl;
```

Muestre la suma de las posiciones del arreglo ‘arr’ usando el apuntador ‘*p’ (*por notación de apuntadores*).

9) Complete el siguiente código C/C++, defina correctamente el parámetro de la Linea#13 y como se debe pasar correctamente ‘cadena’ a f1 en la Linea#21, tal que sobre ‘cadena’ se haga un tratamiento por referencia y se muestre en la Linea#22: *Hola Companero piloso*

```
13 void f1(        ) {
14     strcat(p," piloso");
15 }
16
17
18 int main() {
19     char cadena[32] = "Hola Companero";
20
21     f1(     );
22     cout<<cadena<<endl;
23
24     return 0;
25 }
```

10) Complete el siguiente código C/C++:

- Defina el **1° parámetro** faltante tipo apuntador a función en la función ‘**calcularArea**’ (*Linea#16 en el espacio en blanco antes de la 1° coma*),
- Pase correctamente la variable que corresponda de tipo apuntador a función (*Linea#40 en el espacio en blanco antes de la 1° coma, no use valores literales, siempre variables*)
- y pruebe el correcto funcionamiento del programa:

```
4   using namespace std;
5
6   enum TipoRectangulo { CUADRADO = 0, RECTANGULO = 1};
7
8   double obtenerAreaCuadrado(double base, double altura) {
9       return (pow(base, 2));
10  }
11
12  double obtenerAreaRectangulo(double base, double altura) {
13      return (base * altura);
14  }
15
16  double calcularArea(                                        ),
17                          double base, double altura, TipoRectangulo tiporectangulo) {
18      double resultado = 0;
19
20      resultado = (*rectangulogeneral)(base,altura);
21      switch(tiporectangulo) {
22          case CUADRADO:
23              //logica cuando sea un cuadrado
24              break;
25          case RECTANGULO:
26              //logica cuando sea un rectangulo
27              break;
28          default:
29              //logica cuando no se de ninguno de los anteriores casos
30              break;
31      }
32
33      return resultado;
34  }
35
36  int main() {
37      TipoRectangulo opcion = TipoRectangulo::CUADRADO; //opcion
38      double base = 5.0, altura = 10.0;
39      double(*wrapper[])(double,double) = {&obtenerAreaCuadrado,&obtenerAreaRectangulo};
40      double r = calcularArea(               ,base,altura,opcion);
41
42      cout<<"Area: "<<r<<endl;
43
44      return 0;
45  }
```

- ¿Qué ventaja da el uso de una **enum** (*enumeración*)?

11) Para el siguiente código C/C++, responda:

- ¿Que muestran las líneas 5 y 6?
- En notación de arreglo, ¿Como serían las coordenadas para mostrar el numero 70?

```
4  void f1(int (*p)[2][3]) {
5      cout<<*(*(*p)+8)<<endl;
6      cout<<*(*(*p)+8)+5<<endl;
7  }
8
9  int main() {
10     int nums[2][2][3] = {
11         {
12             {10,20,30},
13             {40,50,60}
14         },
15         {
16             {70,80,90},
17             {100,110,120}
18         }
19     };
20
21     f1(nums);
22
23     return 0;
24 }
25
```

12) Con base al siguiente código C/C++:

- Realice la respectiva representación gráfica correcta, paso a paso de TODO lo que sucede en memoria.
- ¿Qué comportamiento tuvo el 1° caso?
- ¿Qué comportamiento tuvo el 2° caso?

```
4   void f1(int *p, int i) {
5       p = &i;
6   }
7
8   void f2(int **p, int i) {
9       *p = &i;
10  }
11
12  int main() {
13      int a = 100;
14      int *p = NULL;
15
16      p = &a;
17      f1(p,30);
18      cout<<"1er Caso: "<<*p<<endl;
19      f2(&p,30);
20      cout<<"2do Caso: "<<*p<<endl;
21
22      return 0;
23  }
```

13) Con base al siguiente código C/C++:

- En teoría la Linea#14 debería mostrar: 100
  ¿Por qué no muestra 100?
  Algo se debería hacer en la Linea#5, ¿Qué se debería hacer?

```
4   void f1(int **p) {
5       int a = 100;
6       cout<<&a<<endl;
7       *p = &a;
8   }
9
10  int main() {
11      int *p = NULL;
12      f1(&p);
13      cout<<p<<endl;
14      cout<<*p<<endl;
15
16      return 0;
17  }
```