Práctica #1 - Tipos de datos

Parte I - Tipos de datos simples, Tipos de datos estructurados

1. Discuta la veracidad de las siguientes afirmaciones:
   - Un tipo de dato es el conjunto de valores que puede asumir una variable.
   - El uso de tipo de datos incrementa la semántica de un programa y su legibilidad.
   - En los enumerados y en los caracteres existen una relación de orden.
   - Todos los lenguajes implementan el tipo lógico de la misma manera.

2. Explique diferencias entre:
   - Tipos elementales y tipos estructurados.
   - Tipos homogéneos y tipos heterogéneos.

3. Defina una estructura de datos que permita almacenar eventos de un calendario. Esta no debe registrar eventos de fechas inexistentes. Asuma que su calendario funciona para un sólo año. Además se desea conocer el día de la semana en que ocurre el evento.

4. Una empresa de vigilancia privada ha tenido problemas con el comportamiento de sus empleados, por ello decidió ordenar la creación de un sistema que permita almacenar los datos de cada vigilante, de los cuales se desea conocer nombre, CI, dirección, años de experiencia, nacionalidad, país de origen, edad y estado civil. La empresa presta servicios a compañías públicas y privadas; de las públicas se desea conocer el área de servicio que cubren y el número de vigilantes que requieren; de las privadas, la fecha desde la cual se les presta servicio y para ambos tipos nombre, dirección y RIF. Usted debe satisfacer los siguientes requerimientos:

   - Elaborar un algoritmo que permita conocer los datos del vigilante que tiene el mayor número de quejas y en qué compañía.
   - Conocer la compañía que ha formulado el mayor número de quejas.
   - Diseñar una estructura de datos que permita representar esta información.

Parte II - Tipo de Dato Referencia o Apuntadores

5. ¿Cuál es la diferencia entre los operadores & y *?

6. ¿Es posible tener apuntadores a apuntadores y anidarlos cuantas veces sea necesario?

7. Explique las ventajas y desventajas del uso de apuntadores.

8. Para las siguientes instrucciones, construya el estado de todas las variables en la memoria (de forma gráfica) que muestre que ocurre en ella, y cuál es la salida del programa.

```
struct node {
    int info;
    node *next;
};
```

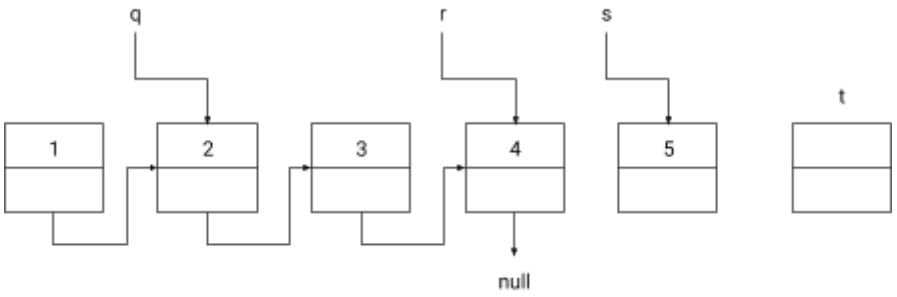
```
1   void main() {
2      node *p, *r, *s;
3      p = new node;
4      s = new node;
5      r = new node;
6      (*p).next = r;
7      (*r).next = s;
8      (*s).next = p;
9      (*s).info = 3;
10     (*((*((*((*p).next)).next)).next)).info = 2;
11     (*((*((*s).next)).next)).info = 1;
12     p = (*s).next;
13     print((*p).info+" "+(*s).info+" "+(*r).info);
14  }
```

9. Dadas las siguientes definiciones:

```
class node {
    public:
        int info;
        node *next;
};
node *q, *r, *s, t;
```

Asuma como estado inicial la figura que a continuación se muestra, e indique el estado final después de ejecutar cada una de las siguientes instrucciones independientemente y luego una tras otra secuencialmente.

```
1   q = (*q).next;
2   *q = *((*q).next);
3   (*q).next;
4   (*((*q).next)).next
5   q = (*r).next;
6   *q = *((*r).next);
7   (*s).next = (*q).next;
8   (*s).next = s;
9   t = *q;
10  *q = *s;
11  *s = t;
12  q = t;
13  (*r).next = q;
14  (*((*((*q).next)).next)).next;
```



10. Dada la siguiente secuencia de instrucciones indique que ocurre en cada línea y señale si queda algún espacio de memoria por liberar.

```
class node {
    public:
        int info;
        node *next;
};
node *p, *q;
int *e, i, **f;

1   void main() {
2      p = new node;
3      e = new int;
4      *e = 0;
5      f = &e;
6      (*p).next = new node;
7      q = &(*p);
8      p = (*p).next;
9      (*q).info = 30;
10     (*p).info = (*q).info + 10;
11     (*p).next = new node;
12     (*((*p).next)).info = (*p).info + (*q).info + 10;
13     p = (*p).next;
14     (*p).next = NULL;
15     while(q != NULL){
16        *e = **f + (*q).info;
17        q = (*q).next;
18     }
19     f = new int*;
20     *f = new int;
21     **f = 1;
```

```
22      delete *f;
23      *f = &i;
24      i = 5;
25      print(**f);
26    }
```

11. Considere las siguientes declaraciones e indique el efecto de las siguientes operaciones:

```
int *x, *y, *z, a;
char *w, b;
bool c;
```

```
1    x = new int;
2    y = new int;
3    w = new char;
4    x = y;
5    b = *w;
6    z = new int;
7    z = w;
8    c = (z == w);
```

```
9    x = new int;
10   *x = 1;
11   *w = 'g';
12   a = *x + *y;
13   c = (a == *w);
14   *z = a;
15   z = x;
16   delete y;
```

12. Dada la siguiente secuencia de instrucciones indique que ocurre en cada línea. Indique además si queda algún elemento por liberar de memoria al terminar recorrido( ).

```
class node {
    public:
        int info;
        node *next;
};
node *p, *q;
```

```
1    void recorrido() {
2       p = new node;
3       (*p).info = 10;
4       (*p).next = p;
5       q = new node;
6       (*p).next = q;
7       (*q).info = (*p).info + 3;
8       (*q).next = p;
9       (*q).next = NULL;
10      q = new node;
11      (*q).info = (*p).info + (*((*p).next)).info;
12      (*((*p).next)).next = q;
13      (*q).next = NULL;
14      q = p;
15      while (q != NULL){
16         print((*q).info);
17         q = (*q).next;
18      }
19    }
```

13. Haga la traza del siguiente algoritmo y explique qué sucede en cada instrucción.

```
1    void main() {
2       int i, j, n, *arr, **mat;
3       read(n);
4       arr = new int[n];
5       mat = new int*[n];
6       for (i = 0; i < n; i++){
7          arr[i] = i;
8          mat[i] = new int[n];
9          for(j = 0; j < n; j++)
10            mat[i][j] = i + j;
```

```
11      }
12      i = n - 1;
13      while (i >= 0){
14         print(arr[i]);
15         delete mat[i];
16         i--;
17      }
18      delete arr;
19      delete mat;          // ¿a quién apunta arr? ¿podría acceder arr[0]?
20    }
```

14. Realice la traza del siguiente algoritmo. En cada línea, indique el estado de las estructuras de datos gráficamente, y en caso de que la instrucción sea incorrecta, señale el tipo de error.

```
struct node {
    int info;
    node *next;
};
node **p, **s, *q, *r;

1   p = &q;                 6   s = &r;
2   (*q).info = 30;         7   delete q;
3   (*q).next = NULL;       8   q = new node;
4   r = new node;           9   (*q).info = 31;
5   (*r).info = 2;          10  (*s).next = q;
                            11  (*(*p)).info = (*q).info + (*((*(*s)).next)).info;
```

15. Haga la traza del siguiente algoritmo y explique qué sucede en cada instrucción.

```
class point {
    public:
        float x, y;
        point(){
            (*this).x = 0;
            (*this).y = 0;
        }
        point(float x, float y){
            (*this).x = x;
            (*this).y = y;
        }
};
class rect{
    public:
        point *p1, *p2;
        rect(){
            (*this).p1 = NULL;
            (*this).p2 = NULL;
        }
        rect(point *p1, point *p2){
            (*this).p1 = p1;
            (*this).p2 = p2;
        }
};

1   void main() {
2      point *a, *b;
3      a = new point;
4      b = new point(1.0f, 1.0f);
5      rect* r = new rect(a, b);
6      delete a;
7      delete r;                    // ¿ocurre un error en esta instrucción?
8   }
```