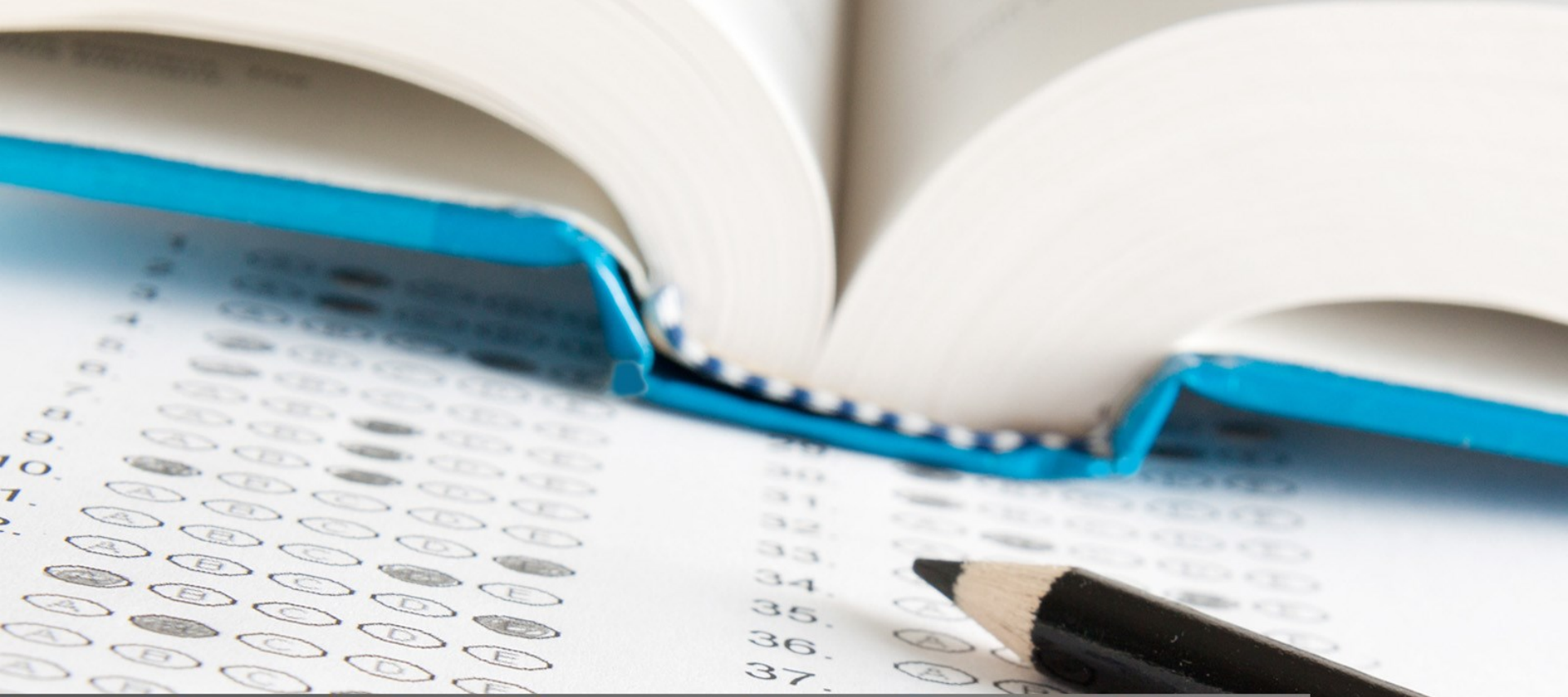

Clase de Repaso – Módulo 1

Taller de Lenguajes I

Ejercicio 1

Complete indicando qué imprimen los siguientes códigos:

a)
```
int i;
for (i = 0; i < 7; i+=1)
        printf("%d ", i++);
```

b)
```
int i;
for (i = 1; i < 9; i*=2)
        printf("%d ", i);
```

c)
```
int i;
for (i = 0; i < 7; i+=1)
        printf("%d ", i);
```

d)
```
int i;
for (i = 4; i > 0; i-=1)
        printf("%d ", i);
```

a) ----------

b) ----------

c) ----------

d) ----------

Ejercicio 2

Complete indicando qué imprime el siguientes código:

```
int main(){
      int i;
      for (i=0;i<6;i++)
          f(i);
      return 0;
  }
void f(int a) {
    static int s = 0;
    if (s++ % 2 == 0)
        printf(" s=%d\n", a);
}
```

Imprime:

.....

.....

.....

Ejercicio 3

Para cada inciso indique si es verdadero o falso.

	a) El operador ++ aplicado a un puntero incrementa la dirección apuntada en uno.
	b) El operador * (de indirección) permite acceder a la variable apuntada por el puntero.
	c) Un arreglo puede manipularse como un puntero.
	d) Un puntero no puede manipularse como un arreglo.
	e) Los punteros en C permiten simular el pasaje por referencia de Pascal.
	f) Un nombre de una variable arreglo sin índice es un puntero al primer elemento del mismo.

Ejercicio 4

Para cada inciso indique si es verdadero o falso.

	a) Para acceder a un campo de una estructura a través de un puntero se utiliza el operador . (punto).
	b) Es posible inicializar los campos de una estructura al momento de definirla.
	c) Dos variables definidas como estructuras pueden compararse si tienen el mismo tipo.
	d) La definición de una estructura no puede almacenar un campo que contenga una matriz.
	e) Es posible declarar una estructura sin asignarle un nombre de tipo asociado.
	f) Es posible declarar un arreglo de punteros a estructuras.

Ejercicio 5

Indique qué imprime el programa. Justifique

```
void leer (int * p);

int main(){
    int * ptr = NULL;
    leer(ptr);
    if (ptr == NULL)
        printf("ptr es NULL\n");
    else
        printf("ptr  es %d\n", *ptr);

    return 0;
}
```

```
void leer (int * p) {

    p = (int *)
            malloc(sizeof(int));
    scanf("%d", p);

}
```

Imprime:

Ejercicio 6

Dado el siguiente código, haga las modificaciones que crea necesarias en caso de que la función no realice su cometido. Indique en la sección de la derecha el número de línea modificada con el código correspondiente.

```
    /* La función concatenar debe agregar la cadena apuntada por texto2 al final de la
    cadena apuntada por texto1. No se verifica que texto1 tenga suficiente espacio para
    almacenar texto2 (esto será responsabilidad del usuario de la función). */
1.  char* concatenar(char* texto1, char* texto2){
2.    int i;
3.    int largo = sizeof(texto1)
4.
5.    for (i=0; i< sizeof(texto2); i++){
6.      texto1[largo + i]= texto2[i];
7.    }
8.
9.    return texto1;
10. }
```


Ejercicio 7

Escriba un programa que:

a) Defina el tipo estructura struct curso, que representa un curso de postgrado de la Facultad. Los datos de los cursos son: código (entero), nombre (texto, 50), duración en horas (entero).

b) Renombre el tipo struct curso a curso_t.

c) Defina la función cargar_curso que inicialice un curso a partir de valores ingresados por teclado.

d) Defina la función prom_duracion que reciba un arreglo de cursos y un entero con su cantidad y retorne el promedio de horas de duración de los cursos.

e) Implemente un programa que: lea desde teclado una cantidad de cursos a ingresar, reserve memoria para estos, cargue los cursos, calcule el promedio, lo imprima en pantalla y libere la memoria.