Ayudantía 4:
Ejercicios Capítulo 3

Profesores: José Luis Martí Lara, Roberto Díaz Urra
Ayudantes:
Hugo Sepúlveda Arriaza
Gabriela Acuña Benito
Lucio Fondón Rebolledo
lucio.fondon@sansano.usm.cl

Universidad Técnica Federico Santa María
Departamento de Informática

Ejercicio 1 (C1 2019-1)

```
1  int main() {
2      int lista[5] = {1, 3, 5, 7, 9};
3      printf("%d", *(lista + 4) + lista[2] + *lista);
4  }
```

Indicar qué salida entrega la ejecución del siguiente código C:

Ejercicio 1 (C1 2019-1)

```c
int main() {
    int lista[5] = {1, 3, 5, 7, 9};
    printf("%d", *(lista + 4) + lista[2] + *lista);
}
```

Indicar qué salida entrega la ejecución del siguiente código C:

R:
Output: 15

Ejercicio 2

```
1  int main(){
2      int ary[4] = {1, 2, 3, 0};
3      int *p = ary + 3;
4      printf("%d%d", p[-2], ary[*p]);
5      return 0;
6  }
```

Indique que muestra el siguiente extracto de código en C.

Ejercicio 2

```
1  int main(){
2      int ary[4] = {1, 2, 3, 0};
3      int *p = ary + 3;
4      printf("%d%d", p[-2], ary[*p]);
5      return 0;
6  }
```

Indique que muestra el siguiente extracto de código en C.

R:
Output: 2 1

Ejercicio 3 (C1 2003-1)

El siguiente código C produce el problema de:

```
1  int foo(){
2      int *p, *q;
3      p = (int *)malloc(sizeof(int));
4      q = p;
5      free(p);
6      *q = 5;
7  }
```

a) Overflow
b) Deadlock
c) Dangling
d) Basura
e) Ninguna de las anteriores

Ejercicio 3 (C1 2003-1)

El siguiente código C produce el problema de:

```
1  int foo(){
2      int *p, *q;
3      p = (int *)malloc(sizeof(int));
4      q = p;
5      free(p);
6      *q = 5;
7  }
```

a) Overflow
b) Deadlock
c) Dangling
d) Basura
e) Ninguna de las anteriores

R: **Alternativa c**

Ejercicio 4 (C1 2009-1)

```
1  int foo(){
2      char *puntero, int p = 100;
3      puntero = malloc(sizeof(char));
4      p = *puntero;
5      p ++;
6      puntero = malloc(sizeof(char));
7      p = *puntero;
8      return p;
9  }
```

Indique que problema tiene el siguiente extracto de código en C.

Ejercicio 4 (C1 2009-1)

```
1  int foo(){
2      char *puntero, int p = 100;
3      puntero = malloc(sizeof(char));
4      p = *puntero;
5      p ++;
6      puntero = malloc(sizeof(char));
7      p = *puntero;
8      return p;
9  }
```

Indique que problema tiene el siguiente extracto de código en C.

R:
Se pierde el puntero generado previamente, es decir, **Basura**.

Ejercicio 5

```
1  int get(int a){
2      int *b = (int *)malloc(sizeof(int));
3      *b = 123;
4      return a++;
5  }
6
7  int main(){
8      int a = 0;
9      while(a < 10) {
10         a = get(a);
11     }
12     return 0;
13 }
```

¿Qué fenómeno ocurre en el siguiente programa C?

Ejercicio 5

```
1  int get(int a){
2      int *b = (int *)malloc(sizeof(int));
3      *b = 123;
4      return a++;
5  }
6
7  int main(){
8      int a = 0;
9      while(a < 10) {
10         a = get(a);
11     }
12     return 0;
13 }
```

¿Qué fenómeno ocurre en el siguiente programa C?

R: Se genera **basura**

Ejercicio 6

Implemente una versión de la función `map()` de Python en el lenguaje de programación C `map(int* A,int n,int(*accion)(int i)`, la cual tomará como parámetros un arreglo `A` de enteros, un entero `n` que será el largo del arreglo y un puntero a una función que se le aplicará al arreglo.

```
map([2,5,3,4],4,duplicar) // [4,10,6,4]
map([2,5,3,4],4,alCuadrado) // [4,25,9,16]
```

Posible Solución

```
void map(int* arr, int n, int(*accion)(int n)){
    printf("Arreglo antes de map(): [");
    for (int i = 0; i < n; ++i){
        printf(" %d ",arr[i]);
    }

    printf("]\n");
    printf("Arreglo después de map(): [");

    for (int i = 0; i < n; ++i){
        arr[i] = accion(arr[i]);
        printf(" %d ",arr[i]);
    }

    printf("]\n");
}

int duplicar(int i){
    return 2*i;
}

int alCuadrado(int i){
    return i*i;
}

int main(){
    int A[] = {1,2,3,4};
    int n = sizeof(A)/sizeof(A[0]);
    printf("Duplicar numeros del arreglo:\n");
    map(A,n,duplicar);
    printf("------------\n");
    printf("Elevar al cuadrado numeros del arreglo:\n");
    map(A,n,alCuadrado);
    return 0;
}
```