2do Parcial

1. Dada una secuencia de n enteros (representada como un arreglo), deseamos encontrar la subsecuencia consecutiva de mayor suma.
El algoritmo obvio consiste en calcular todas las subsecuencias consecutivas posibles, y ver cuál es la de mayor suma. En este ejercicio intentaremos una estrategia alternativa, utilizando programación dinámica.

a) Escriba una fórmula recursiva para la siguiente eureka: $E[k]$ representa la suma de la subsecuencia consecutiva de mayor suma que termina exactamente en la posición k.

b) Describa cómo solucionar el problema original a partir de los valores $E[1], E[2], \ldots, E[n]$.

c) Implemente una función C que solucione el problema dado, utilice el estilo bottom–up.

d) Haga un análisis de complejidad del algoritmo propuesto.

2. La siguiente función representa un algoritmo de ordenamiento.

```
void sort(int data[], int sz) {
  int i, j, z, t;
  for (i = 0; i < sz; i++) {
    z = i;
    for (j = i + 1; j < sz; j++)
      if (data[j] < data[z])
        z = j;
    t = data[i]; data[i] = data[z]; data[z] = t;
  }
  return ;
}
```

a) Identifique el algoritmo de ordenamiento.

b) Describa cuál es el mejor y peor caso del algoritmo sobre un arreglo de tamaño n.

c) Decida si este algoritmo trabaja de manera in–situ (in–place).

d) Escriba una versión recursiva del algoritmo, con el tipo `void rsort(int data[], int sz, int i)`.

e) Escriba la ecuación correspondiente a la complejidad temporal de la función del ítem anterior.

3. El siguiente código representa una versión sin terminar que utiliza la técnica divide et impera para calcular la suma de los elementos de un arreglo de enteros: divide el arreglo en dos mitades, calcula la suma sobre ambas partes y luego las combina.

```
int sum(int data[], int i, int l) {
  int s1, s2;

  if (i == l)
    return ??;

  s1 = ??;
  s2 = ??;

  return s1 + s2;
}
```

a) Completar el código faltante.

b) Hacer un análisis de complejidad del algoritmo.

c) Comparar la complejidad obtenida en el ítem anterior con la del algoritmo obvio para sumar los elementos: iterar entre los elementos, acumulando la suma.

d) Según lo visto en clase, explique la diferencia principal entre programación dinámica y divide et impera, utilice este ejemplo en la explicación.