El problema de la suma de subconjuntos es este: dado un conjunto *a* de *n* enteros, ¿existe algún subconjunto de *a* no vacío cuya suma sea exactamente cero? Por ejemplo, dado el conjunto { −7, −3, −2, 5, 8}, la respuesta es sí, porque el subconjunto { −3, −2, 5} suma cero. Considere 1≤n≤64. La siguiente solución toma tiempo $O(n \cdot 2^n)$.

```
typedef unsigned long long Set;
Set buscar(int a[], int n) {
  Set comb= (1<<(n-1)<<1)-1; // 2^n-1: nº de combinaciones
  for (Set k= 1; k<=comb; k++) {
    // k es el mapa de bits para el subconjunto { a[i] | bit k_i de k es 1 }
    long long sum= 0;
    for (int i= 0; i<n; i++) {
      if ( k & ((Set)1<<i) ) // si bit k_i de k es 1
        sum+= a[i];
    }
    if (sum==0) {   // éxito: el subconjunto suma 0
      return k;     // y el mapa de bits para el subconjunto es k
  } }
  return 0;         // no existe subconjunto que sume 0
}
```

En la función *buscar* los subconjuntos de *a* se representan mediante una máscara de bits $k= k_{n-1} \ldots k_1 k_0$. El elemento *a[i]* está en el subconjunto si k_i es 1. Esta función recorre los 2^n-1 subconjuntos posibles (se descarta el subconjunto vacío) hasta encontrar un subconjunto que sume 0, en cuyo caso se retorna su mapa de bits. Si ningún subconjunto suma 0, se retorna 0.

Reprograme completamente la función *buscar* de modo que la búsqueda se haga paralelamente en 8 cores. Para ello use *fork* para crear 8 procesos pesados. Utilice un *pipe* por cada proceso hijo para que este le entregue al padre el mapa de bits del subconjunto encontrado. Si hay múltiples subconjuntos que suman 0, entregue cualquiera de ellos.

Restricción: si uno de los procesos termina encontrando un subconjunto que suma 0, no espere a que el resto de los procesos termine. Ud. debe matarlos con *kill(pid, SIGTERM)*. Luego entiérrelos y entregue la solución encontrada. Para esperar hasta que algún hijo termine, use:

```
pid_t pid= waitpid(0, NULL, 0);
```

Luego busque *pid* entre los procesos que creó y lea el pipe que abrió para recibir su solución.

Recursos

Baje *t4.zip* de material docente en U-cursos y descomprímalo. El directorio *T4* contiene los archivos *test-suma.c* que prueba si su tarea funciona, *Makefile* que le servirá para compilar su tarea y *suma.h* que contiene el encabezado de la función pedida. Ud. debe programar *buscar* en el archivo *suma.c*. El programa de prueba lo felicitará si su tarea aprueba todos los tests o le indicará cuál test falla.

Evaluación

Entregue su tarea solo si compila y ejecuta sin arrojar warnings en la máquina *anakena.dcc.uchile.cl*. Al invocar *fork* verifique que el resultado es mayor o igual a 0. De otra forma *anakena* fue incapaz de crear el suficiente número de procesos pesados. Inténtelo más tarde.

Entrega

Ud. debe entregar el archivo *suma.c* por medio de U-cursos. Se descontará medio punto por día de atraso. No se consideran los días sábado, domingo, festivos o vacaciones.