Palancas

Tienes un grupo de *N* palancas y todas inicialmente se pueden "bajar". Decimos que una palanca esta activa si no la han bajado e inactiva si la han bajado. Existe una manera de proceder para bajar las palancas y consiste en: Seleccionar una palanca y se baja, después cuentas una palanca activa y la bajas. Luego Cuentas dos palancas activas y bajas la correspondiente y así sucesivamente. Cuando llegas al final de la lista de las palancas sigues tu conteo desde el inicio de la lista de palancas.

Escribe el psedocódigo de una función *F* que recibe como parámetros un entero *N* y un entero *P*. $N > 0$, indica cuantas palancas hay y $P < N$ indica en donde comenzaras el conteo. La función debe de regresar un arreglo *R* de *N* enteros, donde el elemento *R*[i] contiene el valor del número de la palanca que fue bajada en la posición "*i*" en la secuencia. La primera palanca es considerada como la palanca "cero".

Ejemplo:

F(5,1) = {0,1,3,4,2}

F(9,4) = {5,6,8,2,0,1,7,3,4}

Solución

Sea *N* el número de palancas, *P* la palanca inicial a bajar, *M* un arreglo que nos indique las palancas que han sido bajadas, uno si esta activa y cero si esta inactiva (inicialmente todos sus valores son uno) y *R* el arreglo que retornaremos como respuesta.

La función mod(*a*,*b*) regresa el residuo de la división de *a* entre *b*.

```
int * F(int N, int P)
{
Count = 0
Index = P

Mientras Count < N Haz:
        Temp = 0
        Mientras Temp < Count Haz:
                Si M[Index] = 1 Haz:
                        Temp = Temp + 1
                Fin
                Index = mod(Index +1,N)
        Fin
```

```
        R[Count] = Index
        M[Index] = 0
        Count = Count + 1
Fin
}
```

Observemos que la forma de caminar por las palancas, es como situar a las palancas en una mesa redonda. Por tanto, emplearemos "módulos" para tener un índice que camine a lo largo del circulo sin tener que preocuparnos por si llegamos al final. Si tenemos *N* palancas e *Index* es la variable que se mueve en la mesa, entonces para dar un paso sólo tenemos que asignar a *Index* como *Index* + 1 modulo *N*. De éste modo, si *Index* + 1 es igual a *N*, al aplicarle modulo *N*, *Index* + 1 queda como cero y con ello estamos al principio como el problema lo pide.

Notemos que primero tenemos que contar cero palancas activas a partir de la posición inicial *P* que nos hayan dado, luego una palanca activa a partir de donde estábamos parados, luego dos y así sucesivamente. Sea *Count* la variable que aumentará en cada iteración y que nos indica cuantas palancas activas tenemos que contar. Hacemos un primer ciclo de *N* iteraciones en el cual, cada iteración termina cuando contemos *Count* palancas activas. Entonces sea *Temp* inicialmente igual a cero y entramos a un segundo ciclo. En cada iteración del segundo ciclo avanzamos en uno la variable *Index* y preguntamos si donde estamos parados es una palanca activa. De ser así, entonces aumentamos en uno el contador Temp. En caso contrario, no hacemos nada. Repetimos este proceso hasta que *Temp* sea igual a *Count*. Una vez terminado el segundo ciclo, nos fijamos cual es el valor de *Index* y marcamos la palanca, que está en la posición *Index*, como inactiva, aumentamos el contador *Count* en uno y con ello terminamos el primer ciclo.

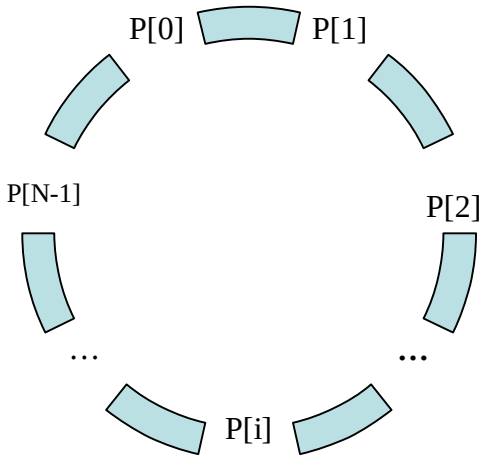

Diagrama de las palancas

Sea N = 9
Entonces los valores de *Index* son:

Si *Index*:	mod(*Index*+1,*N*)
0	1
1	2
2	3
…	
i	*i+1*
…	
N – 2	*N*-1
***N* – 1**	**0**

Código

```
#include "stdio.h"
#include "conio.h"

// Función F
int* F(int N,int P)
{
int *Respuesta = new int[N];        // Arreglo que retornaremos como respuesta.
int *Palancas = new int[N];         // Nos indica si una palanca ha sido bajada
for(int i = 0; i < N; i++)          // Inicializamos
  {
  Respuesta[i] = 0;
  Palancas[i] = 1;
  }

int Count = 0;                      // Count nos indica cuantas palancas llevamos
int Index = P;                      // Index nos indica en que palanca nos encontramos

Respuesta[Count] = Index;           // Inicializamos
Palancas[Index] = 0;
Count++;

while(Count < N)                    // Temp  nos indica cuantas palancas arriba hemos
 {                                  // contado.
 int Temp = 0;
 while( Temp < Count )              // En este loop contamos "Count" palancas que no
  {                                 // han sido bajadas
  Index = (Index + 1)%N;            // Avanzamos
  if( Palancas[Index] == 1 )        // Vemos si la palanca en la posición "Index" esta
   { Temp++; }                      // arriba y aumentamos el contador de palancas
  }

 Palancas[Index] = 0;               // Marcamos la palanca de la posición "Index"
 Respuesta[Count] = Index;          // En la posición "Count" anotamos "index"
 Count++;                           // Aumentamos el contador
 }

return Respuesta;
}

///*************************************************************************
int main()
{
int N,P;                            // Variables empleadas
scanf("%d %d",&N,&P);               // Solicitamos valores a pantalla
```

```
int *Resp = F(N,P);                    // Llamamos a la función F
printf("\n");                          // Imprimimos un salto de línea
for(int i = 0; i < N; i++)             // Imprimimos los resultados
  { printf(" %d ",Resp[i]); }
delete[] Resp;                         // Eliminamos la memoria dinámica
getch();                               // Esperamos

      return 0;
}
```