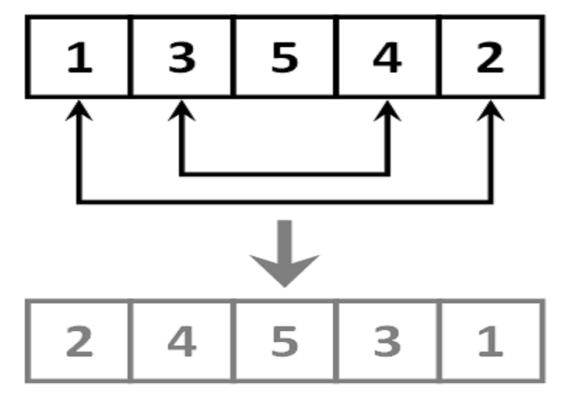
**Nicolás Aguilar León -201530741**

**Juan Sebastián Millán Lejarde – 201516818**

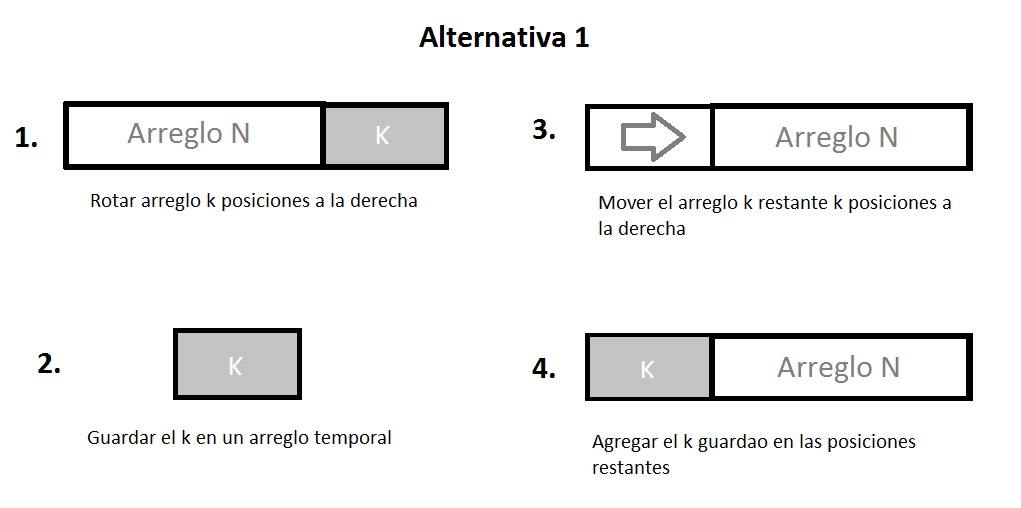
**Algoritmo de solución:**

El algoritmo escogido se seleccionó a partir de la búsqueda de una solución al problema de las rotaciones que minimizara el espacio y tiempo al aplicar n rotaciones en un arreglo. En nuestro caso se decidió utilizar el algoritmo propuesto en un principio por Brian W. Kernighan y P.J. Plauger en su libro de *Software Tools in Pascal*, el cual se basa en declarar una función *Reverse* que se encargue de obtener la inversa del arreglo desde un punto P hasta un Q. Para su utilización en el caso de una rotación se debe aplicar la función 3 veces. La primera obteniendo la inversa del arreglo, partiendo desde uno de mis extremos (La elección de cuál se basa en si se desea rotal el arreglo a la izquierda o a la derecha) hasta la posición de la rotación; la segunda obteniendo la inversa del arreglo restante; y la tercera obteniendo la inversa de todo el arreglo.

La función Reverse funciona de la siguiente manera:

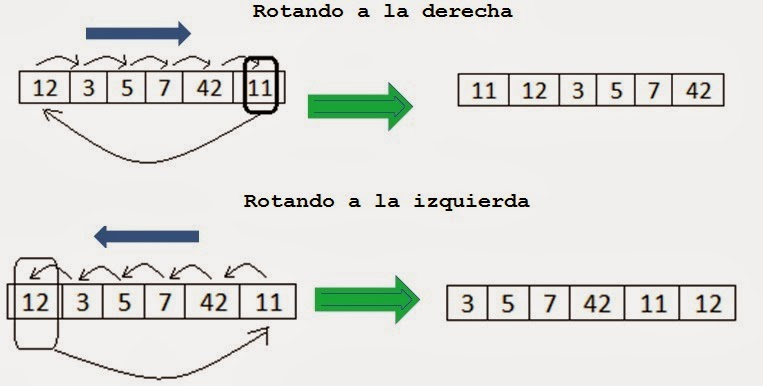


De igual forma, se tuvieron en cuenta otras dos posibles alternativas para resolver el problema. La primera se basaba en utilizar dos arreglos de la siguiente manera: primero debía guardar la primera parte de mi arreglo inicial en un arreglo temporal, luego tendría que mover lo que me queda en mi arreglo K veces, y finalmente tendría que posicionar mi arreglo guardado en las últimas posiciones.



Sin embargo, está solución no es eficiente en espacio por el uso de dos arreglos de tamaño n en lugar de uno como se ve en nuestro caso.

La segunda solución se basaba en crear una función que se encargara de rotar cada posición del arreglo una vez guardando cada posición en una variable temporal y luego llamar a esta función k veces.



Sin embargo, está alternativa no es óptima temporalmente debido a que se debe recorrer cada una de las posiciones y cambiarla de lugar k veces mientras que, en nuestra solución, solo debemos conocer el k que se quiere rotar y utilizar la función del reverso en el arreglo 3 veces.

**Análisis de complejidades espacial y temporal:**

**Espacial:**

La complejidad espacial depende del tamaño del arreglo que se quiere rotar debido a que en él se guardan mis números a los cuales se les quiere aplicar k rotaciones en un definido rango. Sin embargo, si definimos este tamaño en el contexto de nuestro problema, nuestra complejidad espacial se limita a una variable temporal que guarde el valor de una posición para realizar mi función *Reverse.*

**Temporal:**

La complejidad temporal depende del tamaño de mi sub arreglo a modificar multiplicado por el número de rotaciones que se quieren hacer. El tamaño se multiplica por dos debido a que en términos generales se utiliza dos veces la función reverse en el arreglo original. Al resultado se le divide entre dos debido a que el cambio de los valores de lugar se hace modificando dos posiciones en el arreglo. Finalmente, se multiplica por el número de rotaciones que se le quiere hacer al arreglo lo que en el peor caso sería

**Comentarios finales:**

La solución desarrollada tiene un gran desempeño para cualquiera de los casos de prueba que se manejen. En el mejor de los casos, que sería tener un arreglo de cantidad pequeña y aplicar solo una rotación al arreglo de tal forma que la función de *reverse* solo se tenga que aplicar 3 veces para obtener mi resultado, el tiempo que tarda es menor a 0 milisegundos al igual que en el peor de los casos que sería tener un arreglo de tamaño máximo y una alta cantidad de rotaciones que involucren a una gran parte del arreglo.