1) Given the array split the array at the middle (if it is odd length consider next higher integer), add reverse the array.

Ex: if input is [12,10,5,6,52,36] output should be [6,52,36,12,10,5]

if input is [12,10,5,6,52,36,34] output should be [6,52,36,34,12,10,5]

In [56]:

```
import math
# Complete this function to get the desired result
def reverseatCenter(arr):
    n = len(arr)  #length of the array
    m = n//2      #Middle element
    arr2 = arr[m:]+arr[:m]     #Reversing the sub-lists
    return arr2
```

In [57]:

```
# Print wroking examples
print("Example 1 - [14,9,5,6,52,36,34]")
print(reverseatCenter([14,9,5,6,52,36,34,23,111]))
print("Example 2 - [14,9,5,6,52,36]")
print(reverseatCenter([14,9,5,6,52,36]))
```

```
Example 1 - [14,9,5,6,52,36,34]
[52, 36, 34, 23, 111, 14, 9, 5, 6]
Example 2 - [14,9,5,6,52,36]
[6, 52, 36, 14, 9, 5]
```

2) Given a list of numbers, return a list where all adjacent duplicate elements have been removed.

Ex:

2, 2, 2, 3, 2 returns 2, 3, 2.

In [58]:

```
# Complete the function to get the desired result
def remove_adjacent(b):
    b2 = [b[0]]
    for i in range(1,len(b)):
        #Checking consecutive repitions
        if b2[-1]!=b[i]:
            b2.append(b[i])
    return b2
```

In [59]:

```
# Print working examples
print("Example 1 - [2, 2, 2, 3, 2]")
print(remove_adjacent([2, 2, 2, 3, 2]))
print("\n")
print("Example 2 - [1,2,3,3,2,3,1,1,1,2,1,2,2]")
print(remove_adjacent([1,2,3,3,2,3,1,1,1,2,1,2,2]))
```

```
Example 1 - [2, 2, 2, 3, 2]
[2, 3, 2]


Example 2 - [1,2,3,3,2,3,1,1,1,2,1,2,2]
[1, 2, 3, 2, 3, 1, 2, 1, 2]
```

3) given matrix of 7x7 full of ones, create a square with given side length (center same as original square(7x7)) replace ones with zeors at the edges for example,

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

After modification

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

In [60]:

```
import numpy as np
# write programm to get the desired result
def matrix_square(A,l):
    n = A.shape[0]  #Size of the matrix
    #the length of the smaller square should be lesser than the size of the matrix, bot
h of which
    #are odd
    assert l<=n and n%2==1 and l%2==1

    #start and end of inner square
    start = (n-l)//2
    end = (n+l)//2

    for i in range(start,end):
        for j in range(start,end):
            #Making the elements in the edges of the smaller square 0
            if (i==start or i==end-1) or (j==start or j==end-1):
                A[i][j] = 0


    return A.astype(int)
```

In [61]:

```
#Print working examples

#Example 1 - side length 3
print("Example 1 - side length 3")
A = np.ones((7,7))  #7x7 matrix of ones
l = 3  #Side length = 3
print(matrix_square(A,l))
print("\n")

#Example 2 - side length 5
print("Example 2 - side length 5")
A = np.ones((7,7))  #7x7 matrix of ones
l = 5  #Side length = 5
print(matrix_square(A,l))
print("\n")

#Example 3 - side length 7
print("Example 3 - side length 7")
A = np.ones((7,7))  #7x7 matrix of ones
l = 7  #Side length = 7
print(matrix_square(A,l))
```

```
Example 1 - side length 3
[[1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1]
 [1 1 0 0 0 1 1]
 [1 1 0 1 0 1 1]
 [1 1 0 0 0 1 1]
 [1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1]]


Example 2 - side length 5
[[1 1 1 1 1 1 1]
 [1 0 0 0 0 0 1]
 [1 0 1 1 1 0 1]
 [1 0 1 1 1 0 1]
 [1 0 1 1 1 0 1]
 [1 0 0 0 0 0 1]
 [1 1 1 1 1 1 1]]


Example 3 - side length 7
[[0 0 0 0 0 0 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 1 1 1 1 1 0]
 [0 0 0 0 0 0 0]]
```

4) Paragraph present in data.txt is encoded such that each alphabet in word is incremented to next ascii value. Decode the paragraph present in the data_encoded.txt (Hint: decrease the ascii value of each character in the word)

In [62]:

```
def sentence_decode(str):

    dec_words = ""   #initializing empty string

    for character in str:
        #Copying spaces as it is
        if character==" ":
            dec_words+=" "
        #Decoding - identifying the previous ascii value
        else:
            dec_words += chr(ord(character)-1)
    return dec_words
```

In [63]:

```
with open ("data_encoded.txt", "r") as myfile:
    data=myfile.read()
    print(data)
    print(" ")
    print(sentence_decode(data))
```

```
ebub bobmztjt jt b qspdftt pg jotqfdujoh- dmfbotjoh- usbotgpsnjoh boe npef
mjoh ebub xjui uif hpbm pg ejtdpwfsjoh vtfgvm jogpsnbujpo- jogpsnjoh dpodm
vtjpot boe tvqqpsujoh efdjtjpo.nbljoh/ ebub bobmztjt ibt nvmujqmf gbdfut b
oe bqqspbdift- fodpnqbttjoh ejwfstf ufdiojrvft voefs b wbsjfuz pg obnft- b
oe jt vtfe jo ejggfsfou cvtjoftt- tdjfodf- boe tpdjbm tdjfodf epnbjot/ jo
upebz(t cvtjoftt xpsme- ebub bobmztjt qmbzt b spmf jo nbljoh efdjtjpot nps
f tdjfoujgjd boe ifmqjoh cvtjofttft pqfsbuf npsf fggfdujwfmz/

data analysis is a process of inspecting, cleansing, transforming and mode
ling data with the goal of discovering useful information, informing concl
usions and supporting decision-making. data analysis has multiple facets a
nd approaches, encompassing diverse techniques under a variety of names, a
nd is used in different business, science, and social science domains. in
today's business world, data analysis plays a role in making decisions mor
e scientific and helping businesses operate more effectively.
```

In [48]:

```
# Has to print the decoded paragraph.
```