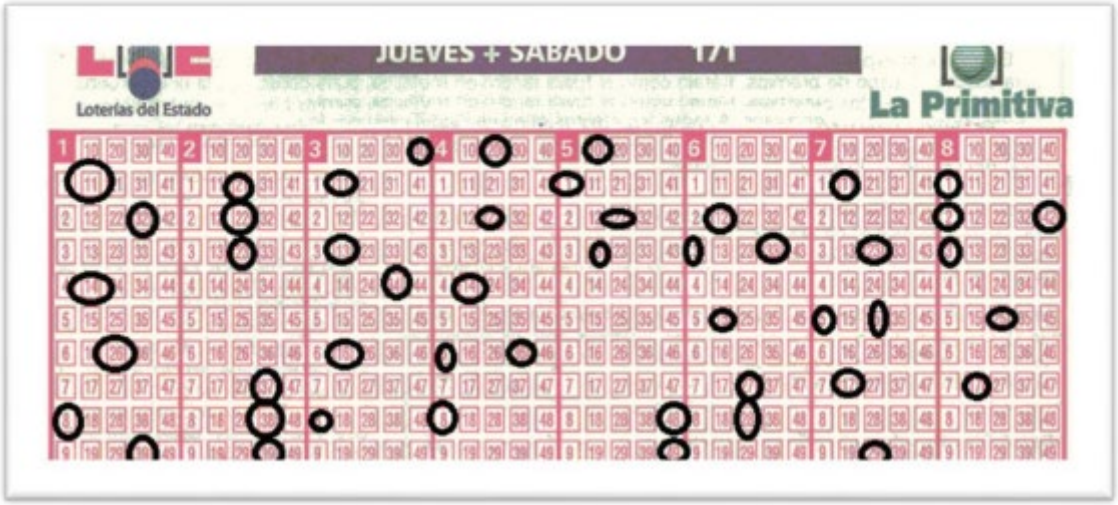
Programe una función recursiva en Python que calcule el número de “triunfos” de que hay en una baraja española de 40 cartas (array de cartas) entre dos posiciones. Las cartas son registros con palo (oros, copas, espadas o bastos) y valor (números enteros del 1 al 7 y del 10 al 12), y se consideran “triunfos” el as, el 3, la sota (10), el caballo (11) y el rey (12).

```
def suma_triunfos(baraja, inicio, fin):
 """ lista, int, int, string -> int
 OBJ: Determina el número de cartas "triunfo" (as,3,10,11 y 12)
 entre 2 límites [inicio..fin] en una baraja de cartas
 PRE: Existe un diccionario "Carta" con dos campos: palo y valor
 """
       if inicio > fin:
            resultado = 0
      else:
            if baraja[0]['valor'] in [1,3,10,11,12]:
                  resultado = 1 + suma_triunfos(baraja, inicio+1, fin)
            else:
                   resultado = suma_triunfos(baraja, inicio+1, fin)
 return resultado
```



En el ejemplo mostrado en la figura, si la combinación ganadora hubiera sido 8-11- 13-16-34-43, su módulo habría determinado que se trata de un boleto ganador de 5 aciertos, pues como se ve, la tercera apuesta incluye 5 de los números de la combinación ganadora.

```
def aciertos(apuesta,combinacion_ganadora):
       aciertos = 0
      for numero in apuesta:
             if numero in combinacion_ganadora:
                  aciertos += 1
      return aciertos

#otra versión (recursiva)
def aciertos_rec(apuesta,combinacion_ganadora):
    if apuesta==[]:
        res=0
    else:
        if apuesta[0] not in combinacion_ganadora:
```

```
            res=aciertos_rec(apuesta[1:],combinacion_ganadora)
        else:
            res=1+ aciertos_rec(apuesta[1:],combinacion_ganadora)
    return res


def maximo_premio (lista_apuestas,combinacion_ganadora):
 """ lista, lista -> bool
 OBJ: Determina el máximo número de aciertos de una apuesta
 de la lotería primitiva.
 PRE: apuestas y comb. ganadora ordenados ascendentemente """
      if lista_apuestas == []:
            resultado = 0
      else:
            aciertos_actual=aciertos(lista_apuestas[0],combinacion_ganadora)
            if aciertos_actual == 0:
                  resultado = 0
            else:
                  resultado = max(aciertos_actual,\

      maximo_premio(lista_apuestas[1:],combinacion_ganadora))
      return resultado
```

La Federación Española de Fútbol registra todos los jugadores de Primera División en una lista, incluyendo varios datos de cada uno tales como la posición en el campo (defensa, medio o delantero), el equipo en que milita, o el número de goles marcados en la temporada. Cada año la Federación organiza su gala de premios con diferentes categorías. Para ser candidato a la categoría de mejor delantero es necesario jugar en esa posición y haber marcado más de 20 goles. Se pide: a) Declarar en Python las estructuras necesarias para gestionar la información almacenada (1 punto)

SOLUCIÓN:

```
lista_federacion = []
# ejemplo de introducción de jugador
jugador = {'nombre':'Messi', 'goles':25,'posicion':'delantero','club':
'FCBarcelona'}
lista_federacion.append(jugador)
jugador = {'nombre':'Suarez', 'goles':30,'posicion':'delantero','club'
:'FCBarcelona'}
lista_federacion.append(jugador)
jugador = {'nombre':'Benzema', 'goles':27,'posicion':'delantero','club
':'RMadrid'}
lista_federacion.append(jugador)
jugador = {'nombre':'Rodrygo', 'goles':31,'posicion':'delantero','club
':'RMadrid'}
lista_federacion.append(jugador)
jugador = {'nombre':'Bale', 'goles':25,'posicion':'delantero','club':'
RMadrid'}
lista_federacion.append(jugador)
jugador = {'nombre':'Morata', 'goles':22,'posicion':'delantero','club'
:'ATMAdrid'}
lista_federacion.append(jugador)
```

b) Programar una función recursiva que a partir de la información que guarda la Federación sobre todos los jugadores, indique cuántos jugadores de Primera División son candidatos al premio al mejor delantero. (2 ptos)

SOLUCIÓN:

```
def candidatos(lista_jugadores, inicio, fin):
    """ list, int, int -> int
    OBJ: Computa el número de candidatos a mejor delantero
```

```
    """
    if inicio > fin:
        resultado = 0
    else:
        if (lista_jugadores[inicio]['posicion'] == "delantero" and \
            lista_jugadores[inicio]['goles'] >= 20):
            resultado = 1 + candidatos(lista_jugadores, inicio+1, fin)
        else:
            resultado = candidatos(lista_jugadores, inicio+1, fin)
    return resultado

print(candidatos(lista_federacion,0,len(lista_federacion)-1))
```

c) Programar un procedimiento que muestre en pantalla todos los clubes de fútbol que tienen al menos un candidato en la competición al mejor delantero, ordenados descendentemente de más a menos candidatos. La salida similar a la siguiente (2 puntos):
FCBarcelona – 3 delanteros
Real Madrid – 2 delanteros
ATMadrid – 2 delanteros
Sevilla FC – 1 delantero
Valencia FC – 1 delantero

SOLUCIÓN:

```
def es_candidato_a_premio(jugador):
    """ jugador --> bool
    OBJ: Averigua si un jugador es candidato"""

    return jugador['goles'] > 20 and jugador['posicion']=='delantero'

def obtener_equipos_con_candidatos(lista):
    """ list -> dictionary
    OBJ: Crear un diccionario con los clubes que tienen al menos un
candidato """
    clubes_premio = {}
    for jugador in lista:
        if es_candidato_a_premio(jugador):
            if jugador['club'] not in clubes_premio:
                clubes_premio[jugador['club']] = 1
            else:
                clubes_premio[jugador['club']] += 1
    return clubes_premio

def es_menor(equipo1, equipo2):
    """ list -> bool
    OBJ: Compara si son menores los elems de la pos 1 de dos listas"""
    return equipo1[1] < equipo2[1]

def ascender(v,inicio,fin):
    for i in range (fin,inicio,-1):
        if es_menor(v[i],v[i-1]):
            temp = v[i]
            v[i] = v[i-1]
            v[i-1] = temp
    return

def burbuja (v,inicio,fin):
    for pasada in range (inicio,fin) :
        ascender(v,pasada,fin)
```

```
def obtener_candidatos_ordenados(lista_federacion):
    """ list -> None
        OBJ: Obtiene todos los clubes de fútbol que tienen al menos un
      candidato en la    competición al mejor delantero, ordenados
      dedescendentemente de más a    menos candidatos"""

    candidatos = obtener_equipos_con_candidatos(lista_federacion)
    #transformamos el diccionario en una lista, para ordenar los datos
    lista_candidatos = []
    for equipo in candidatos:
        lista_candidatos.append([equipo,candidatos[equipo]])
       # Aplicamos el método de ordenación burbuja a la lista, usando

       # como criterio de ordenación el número de candidatos por
equipo
        burbuja(lista_candidatos,0,len(lista_candidatos)-1)
        lista_candidatos.reverse()
    return lista_candidatos


lista_candidatos=mostrar_candidatos_ordenados(lista_federacion)
for equipo in lista_candidatos:
        print(equipo[0],'-', equipo[1],'candidatos')
```