Question 1 – Il s'agit de la commande `len(C)`.

Question 2.a –

```
def appartient(L, x, i0):
    """appartient(L: list, x, i0: int) -> bool"""
    assert 0 <= i0
    for i in range(i0, len(L)):
        if x == L[i]:
            return True
    return False
```

Question 2.b –

```
def tousDiff(C):
    """tousDiff(C: list[str]) -> bool"""
    for i in range(len(C)-1):
        if appartient(C, C[i], i+1):
            return False
    return True
```

Question 3.a –

```
def nbOcc(V,s):
    """nbOcc(V: list[str],s: str) -> int"""
    res = 0
    for e in V:
        if e == s:
            res += 1
    return res
```

Question 3.b –

```
def makeR(C,V):
    """makeR(C: list[str],V: list[str]) -> list[int]"""
    R = []
    for s in C:
        R.append(nbOcc(V, s))
    return R
```

Question 4.a –

```
def nomToNum(C, s):
    """nomToNum(C: list[str], s: str) -> int ou NoneType"""
    for i in range(len(C)):
        if C[i] == s:
            return i
    return None
```

Question 4.b –

```
def makeRBis(C,V):
    """makeRBis(C: list[str],V: list[str]) -> list[int]"""
    R = [0]*len(C)
    for s in V:
        i = nomToNum(C, s)
        if i is not None:  # OU i != None
            R[i] += 1
    return R
```

Question 5.a –

```
def nbVotes(R):
    """nbVotes(R: list[int]) -> int"""
    res = 0
    for e in R:
        res += e
    return res
```

Question 5.b –

```
def majAbs(R):
    """majAbs(R: list[int]) -> int"""
    return nbVotes(R)//2 + 1
```

Question 5.c – Voir les question précédentes.

Question 6.a –

```
def vainqueurTour1(R):
    """vainqueurTour1(R: list[int]) -> int ou NoneType"""
    seuil = majAbs(R)
    for i in range(len(R)):
        if R[i] >= seuil:
            return i
    return None
```

Question 6.b –

```
def deuxMeilleurs(R):
    """deuxMeilleurs(R: list[int]) -> (int, int)"""
    if R[0] >= R[1]:
        i1 = 0
        i2 = 1
    else:
        i1 = 1
        i2 = 0
    for i in range(2,len(R)):
        if R[i] > R[i1]:
            i2 = i1
            i1 = i
        elif R[i] > R[i2]:
            i2 = i
    return (i1,i2)
```

Question 7.a –

```
def valAbs(a):
    """valAbs(a: int) -> int"""
    if a >= 0:
        return a
    else:
        return -a
```

Question 7.b –

```
# nv1 = Nombre de voix qu'obtient i1 au 2nd tour
# nv2 = Nombre de voix qu'obtient i2 au 2nd tour
# La fonction nb_voix calcule nv1 et nv2
def nb_voix(R, i1, i2):
    """nb_voix(R: list[int], i1: int, i2: int) -> (int, int)"""
    nv1 = 0
    nv2 = 0
    for i in range(len(R)):
        if valAbs(i - i1) <= valAbs(i - i2):
            nv1 += R[i]
        else:
            nv2 += R[i]
    return nv1, nv2
```

```
def vainqueurTour2(R, i1, i2):
    """vainqueurTour2(R: list[int], i1: int, i2: int) -> int"""
    nv1, nv2 = nb_voix(R, i1, i2)
    if nv1 > nv2:
        return i1
    elif nv1 < nv2:
        return i2
    else:
        if i1 < i2:
            return i1
        else:
            return i2
```

Question 8 –

```
def vainqueurElection(C, V):
    """vainqueurElection(C: list[str], V: list[str]) -> str ou NoneType"""
    if not tousDiff(C):
        return None
    R = makeR(C,V)
    elu_tour1 = vainqueurTour1(R)
    if elu_tour1 is not None:  # OU elu_tour1 != None
        return C[elu_tour1]
    (i1,i2) = deuxMeilleurs(R)
    return C[vainqueurTour2(R, i1, i2)]
```

Question 9 – Il faut d'abord importer le module `random` en écrivant « `import random` » au début du fichier. On peut alors utiliser la fonction en écrivant `random.randint`.

Question 10.a – La signature de la fonction `f` est :

`f(m: int) -> str`

Cette fonction renvoie une chaîne de caractères `s` de taille `m` où chaque caractère a été choisi aléatoirement parmi les 26 lettres minuscules de l'alphabet.

Question 10.b – La signature de la fonction `g` est :

`g(n: int) -> list[str]`

Cette fonction renvoie une liste `C` contenant `n` chaînes de caractères toutes différentes. Chaque élément de `C` est généré par la fonction `f` (voir question précédente) et sa longueur est choisie aléatoirement dans l'intervalle $[\![4, 10]\!]$.

Question 11 –

```
def listeAlea1(m, s):
    """listeAlea1(m: int, s: int) -> list[int]"""
    A = []
    for _ in range(m-1):
        r = random.randint(0, s)
        A.append(r)
        s = s-r
    A.append(s)
    return A
```

Question 12 –

⋆ Solution 1 : on construit une liste de booléens représentant le tableau de l'énoncé.

```
# On appelle T le tableau décrit dans l'énoncé.
# T[i] vaut False si la case est vide et True si elle contient une croix
def getT(m, s):
    """getT(m: int, s: int) -> list[bool]"""
    T = [False]*(s+m-1)
    nb_croix = 0
    while nb_croix < m-1:
        i = random.randint(0, s+m-2)
        if not T[i]:
            T[i] = True
            nb_croix += 1
    return T
```

```
# Cette fonction renvoie le nombre de False consécutifs se trouvant
# dans T après l'indice i0 (i0 est inclus).
def nbFalse(T, i0):
    """nbFalse(T: list[bool], i0: int) -> int"""
    nb = 0
    for i in range(i0, len(T)):
        if T[i]:
            break
        else:
            nb += 1
    return nb
```

```
def listeAlea2(m, s):
    """listeAlea2(m: int, s: int) -> list[int]"""
    T = getT(m, s)
    A = []
    i = 0
    for _ in range(m):
        a = nbFalse(T, i)
        A.append(a)
        i += a+1
    return A
```

⋆ Solution 2 (plus efficace) : on construit une liste (triée !) contenant les indices des croix dans le tableau.

```
def mini(L):
    """mini(L: list[int]) -> int"""
    i_min = 0
    for i in range(1, len(L)):
        if L[i] < L[i_min]:
            i_min = i
    return i_min
```

```
def tri(L):
    """tri(L: list[int]) -> list[int]"""
    M = []
    while len(L) > 0:
        i_min = mini(L)
        M.append(L[i_min])
        L = L[:i_min] + L[i_min+1:]
    return M
```

```
def pos_croix(m, s):
    """pos_croix(m: int, s: int) -> list[int]"""
    pos = []
    while len(pos) < m-1:
        i = random.randint(0, s+m-2)
        if not appartient(pos, i, 0):
            pos.append(i)
    return tri(pos)
```

```
def listeAlea2(m, s):
    """listeAlea2(m: int, s: int) -> list[int]"""
    if m == 1:
        return [s]
    pos = pos_croix(m, s)
    A = [pos[0]]
    for i in range(len(pos)-1):
        A.append(pos[i+1] - pos[i] - 1)
    A.append(s+m-2-pos[-1])
    return A
```

Question 13 –

```
def genererR(n, s):
    """genererR(n: int, s: int) -> (list[int], int, int)"""
    m = n+2
    L = listeAlea2(m, s)
    R = L[:-2]
    v1 = L[-2]
    v2 = L[-1]
    return (R, v1, v2)
```

Question 14 –

```
# On fait en sorte de ne pas modifier la liste donnée en entrée.
def melange(L0):
    """melange(L0: list) -> list"""
    L = L0[:]
    n = len(L)
    for i in range(n-1, 0, -1):
        j = random.randint(0,i)
        tmp = L[i]
        L[i] = L[j]
        L[j] = tmp
    return L
```

Question 15 –

```
def genererV(C, R, v1, v2):
    """genererV(C: list[str], R: list[int], v1: int, v2: int) -> list[str]"""
    V0 = []
    for i in range(len(C)):
        L = [C[i]] * R[i]
        V0 = V0 + L
    V0 = V0 + [""]*v1
    for _ in range(v2):
        V0.append(f(11))
    return melange(V0)
```

Question 16 –

```
def test(n, s):
    """test(n: int, s: int) -> bool"""
    C = g(n)
    (R, v1, v2) = genererR(n, s)
    V = genererV(C, R, v1, v2)
    return makeR(C,V) == R and makeRBis(C,V) == R
```