Hitza taulan dagoela egiaztatzeko, bi funtzio osatu ditut *BoggleJokatu* klasean. *taulanDago(String)* funtzioak taulako elementuak korritzen ditu, eta *hemendikJarraitzenDu(String, int, int)* funtzioari deitzen dio hitza posizio jakin batean dagoen jakiteko.

taulanDago(String)

Hasteko, balio boolearra itzultzen duen *taulanDago(String)* funtzioa sortu dut. Metodo honek taularen elementu guztiak korrituko ditu, eta *true* itzuliko du bi baldintza hauek betetzen direnean:

- Posizio jakin bateko elementua eta hitzaren lehenengo karakterea berdina izatea.
- hemendikJarraitzenDu(String, int, int) funtzioak true itzultzea, hau da, posizio jakin horretatik abiatuz hitza lortzea posiblea bada.

Taulako elementu guztiak korritu eta bakar batek ere ez baditu bi baldintza horiek bete, false itzuliko du, hau da, ez da aurkitu hitza taulan.

hemendikJarraitzenDu(String, int, int)

Hitz osoa taulan dagoela egiaztatzeko, *hemendikJarraitzenDu(String, int, int)* funtzioak hitza eta bere lehenengo karakterearen posizioa jasoko ditu. Datu horiek erabiliko ditu zer egin behar duen erabakitzeko. Hona hemen bi kasu nabariak:

- 1. Hitza jasotako posizio horretan ez dagoela egiaztatzen duena: Hitzaren lehenengo karakterea ez badago jasotako posizioan, *false* itzuliko du.

 Adibidez: *hemendikJarraitzenDu("ab", 0, 0)* eginez, 'a' karakterea ez badago taularen [0][0] posizioan, *false* itzuliko du.
- 2. Hitza taulan dagoela egiaztatzen duena: Hitzak karaktere bakar bat badu eta karaktere hori funtzioak jasotako posizioan badago, hitza aurkitu dela esan nahi du, eta *true* itzuliko du.

Adibidez: hemendikJarraitzenDu("a", 0, 0) eginez, 'a' karakterea taularen [0][0] posizioan badago true itzuliko du.

Hori dela eta, karaktere bakarreko hitza jasotzean *true* edo *false* itzuliko da beti. Karaktere gehiago dituzten hitzak taulan daudela egiaztatzeko, ordea, hitzaren karaktereak banan-banan irakurri beharko dira, eta taulan bata bestearen ondoan daudela egiaztatu.

Gainera, kontuan izan behar da posizioaren arabera taularen zenbait posizio ezingo direla atzitu. Izan ere, hemendikJarraitzenDu("ab", 0, 0) badugu eta 'a' karakterea [0][0] posizioan badago, 'b' karakterea bilatu beharko dugu [0][1], [1][0] eta [1][1] posizioetan bakarrik, beste 5 posizioak taularen mugetatik kanpo baitaude: [-1][-1], [-1][0], [-1][1], [0][-1] eta [1][-1].

Hori dela eta, prozesu hau modu errekurtsiboan egingo dela erabaki dut, eta taula 9 zatitan banatuta balego bezala idatzi dut kodea. Behatzen ari garen karakterearen posizioaren arabera, taularen posizio jakin batzuk atzituko dira:

- Izkinak: aldameneko 3 posizioak. Adibidez: [0][0] => [0][1], [1][0] eta [1][1].
- Aldeak: aldameneko 5 posizioak. Adibidez: [0][1] => [0][0], [0][2], [1][0], [1][1] eta [1][2].
- Erdiko zatiak: inguruan dauden 8 posizioak.

Hona hemen taularen 9 zatiak, irudikatuta:

2. Zein da j?

		j == 0	bestela	j == m-1
	i == 0	[0][0]	[0][1m-2]	[0][m-1]
1. Zein da i?	bestela	[1n-2][0]	[1n-2][1m-2]	[1n-2][m-1]
	i == n-1	[n-1][0]	[n-1][1m-2]	[n-1][m-1]

Balio boolearrak itzultzen direnez, || (or) eragiketa logikoaren bitartez elkartzen dira funtzioaren emaitzak. Horrela, nahikoa izango da dei bakar batek egiaztatzen badu hitza taulan dagoela.

Dei errekurtsibo hauek orden jakin batean burutzen dira: 3 dei egiten badira eta lehenengoak *true* itzultzen badu, beste bi deiak ez dira exekutatu beharko.

Adibidez: honako taula hau erabilita, "dado" hitza bilatuko dugu:

z i a d +	taulanDago("dado"): taulako elementuak korritzen hasiko da errenkadaka, 'd' karakterea aurkitu arte.
z i a d r d e o t t k a g a a k	'd' karakterea aurkitu ondoren, hemendikJarraitzenDu("dado", 0, 3) funtzioak 'a' karakterea bilatuko du aldameneko posizioetan, ordena honetan: hemendikJarraitzenDu("ado", 0, 2) hemendikJarraitzenDu("ado", 1, 2) hemendikJarraitzenDu("ado", 1, 3)

	z	i	a	d
1	r	d	e	0
1	i	t	k	a
1	g	a	a	k

'a' karakterea aurkitu ondoren, hemendikJarraitzenDu("ado", 0, 2) funtzioak 'd' karakterea bilatuko du aldameneko posizioetan, ordena honetan:

hemendikJarraitzenDu("do", 0, 1)

hemendikJarraitzenDu("do", 0, 3)

hemendikJarraitzenDu("do", 1, 1)

hemendikJarraitzenDu("do", 1, 2)

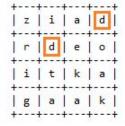
hemendikJarraitzenDu("do", 1, 3)

'd' karakterea aurkitu ondoren, hemendikJarraitzenDu("do", 0, 3) funtzioak 'o' karakterea bilatuko du aldameneko posizioetan, ordena honetan:

hemendikJarraitzenDu("o", 0, 2)

hemendikJarraitzenDu("o", 1, 2)

hemendikJarraitzenDu("o", 1, 3)



Hiru funtzio horietatik gutxienez batek *true* itzultzen badu, hemendikJarraitzenDu("do", 0, 3) funtzioak ere *true* itzuliko du. Bestela, hemendikJarraitzenDu("do", 1, 1) funtzioa exekutatuko da, eta 'o' karakterea bilatuko du aldameneko posizioetan, ordena honetan:

hemendikJarraitzenDu("o", 0, 0)

hemendikJarraitzenDu("o", 0, 1)

hemendikJarraitzenDu("o", 0, 2)

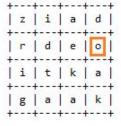
hemendikJarraitzenDu("o", 1, 0)

hemendikJarraitzenDu("o", 1, 2)

hemendikJarraitzenDu("o", 2, 0)

hemendikJarraitzenDu("o", 2, 1)

hemendikJarraitzenDu("o", 2, 2)



Azkenean, 'o' karakterea aurkitu eta hitzaren azkeneko karakterea denez hemendikJarraitzenDu("o", 1, 3) funtzioak *true* itzuliko du. Balio hori hemendikJarraitzenDu("do", 0, 3) deiak jasoko du, eta honek ere *true* itzuliko du. Berdina egingo dute hemendikJarraitzenDu("ado", 0, 2) eta hemendikJarraitzenDu("dado", 0, 3) deiek. Azkenean, hasieran deitutako taulanDago("dado") funtzioak ere *true* itzuliko du, eta horri esker jakingo dugu "dado" hitza taulan dagoela.

hemendikJarraitzenDu(String, int, int) exekutatzen den bakoitzean hitza eta itzultzen duen balioa idatziz gero, honako hau lortuko genuke:

- 1. <u>hemendikJarraitzenDu("dado", 0, 3): dado</u> <u>Dei errekurtsiboa:</u>
 - 1.1. hemendikJarraitzenDu("ado", 0, 2): ado Dei errekurtsiboak:
 - 1.1.1. <u>hemendikJarraitzenDu("do", 0, 1): do</u> *false* itzuli
 - 1.1.2. <u>hemendikJarraitzenDu("do", 0, 3): do</u>
 <u>Dei errekurtsiboak:</u>
 - 1.1.2.1. hemendikJarraitzenDu("o", 0, 2): o false itzuli
 - 1.1.2.2. hemendikJarraitzenDu("o", 1, 2): o false itzuli
 - 1.1.2.3. hemendikJarraitzenDu("o", 1, 3): o *true* itzuli

true itzuli

true itzuli

true itzuli