AlgoritmoPrincipal Caso Normal 1

calculateX: AreaUnderTDistribution: Caso normal	p = 0.2; dof = 6;						
Instruction	**abs(dTargetP - dP) > dErro**	**dTargetP**	**dP**	**dError**	**dD**	**dX**	**dPreviousError**
declare dTargetP and initialise with dP		0.2	0.20000000				
declare dError and initialise with 0.00000001				0.00000001			
declare dD and initialise with 0.5					0.50000000000		
assign 1.0 to dX						1.000000000000000000000000	
call calculate()			0.32204100				
declare dPreviousError and initialise with dTargetP - dP							-0.12204100
while abs(dTargetP - dP) > dError	TRUE	0.2	0.32204100	0.00000001	0.50000000000	1.000000000000000000000000	0.20000000
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.500000000000000000000000	
call calculate()			0.18256000				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.25000000000		
dPreviousError = dTargetP - dP							0.01744000
while abs(dTargetP - dP) > dError	TRUE	0.2	0.18256000	0.00000001	0.25000000000	0.500000000000000000000000	0.01744000
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.750000000000000000000000	
call calculate()			0.27289500				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.12500000000		
dPreviousError = dTargetP - dP							-0.07289500
while abs(dTargetP - dP) > dError	TRUE	0.2	0.27289500	0.00000001	0.12500000000	0.750000000000000000000000	-0.07289500
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.625000000000000000000000	
call calculate()			0.21477200				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.12500000000		
dPreviousError = dTargetP - dP							-0.01477200
while abs(dTargetP - dP) > dError	TRUE	0.2	0.21477200	0.00000001	0.12500000000	0.625000000000000000000000	-0.01477200
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.500000000000000000000000	
call calculate()			0.18256000				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.06250000000		
dPreviousError = dTargetP - dP							0.01744000
while abs(dTargetP - dP) > dError	TRUE	0.2	0.18256000	0.00000001	0.06250000000	0.500000000000000000000000	0.01744000
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.562500000000000000000000	
call calculate()			0.20292400				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.03125000000		
dPreviousError = dTargetP - dP							-0.00292400
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20292400	0.00000001	0.03125000000	0.562500000000000000000000	-0.00292400

calculateX: AreaUnderTDistribution: Caso normal	p = 0.2; dof = 6;						
Instruction	abs(dTargetP - dP) > dErro	dTargetP	dP	dError	dD	dX	dPreviousError
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5312500000000000000000	
call calculate()			0.19283600				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.01562500000		
dPreviousError = dTargetP - dP							0.00716400
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19283600	0.00000001	0.01562500000	0.5312500000000000000000	0.00716400
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5468750000000000000000	
call calculate()			0.19790400				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.01562500000		
dPreviousError = dTargetP - dP							0.00209600
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19790400	0.00000001	0.01562500000	0.5468750000000000000000	0.00209600
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5625000000000000000000	
call calculate()			0.20292400				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00781250000		
dPreviousError = dTargetP - dP							-0.00292400
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20292400	0.00000001	0.00781250000	0.5625000000000000000000	-0.00292400
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5546875000000000000000	
call calculate()			0.20042000				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00781250000		
dPreviousError = dTargetP - dP							-0.00042000
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20042000	0.00000001	0.00781250000	0.5546875000000000000000	-0.00042000
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5468750000000000000000	
call calculate()			0.19790400				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00390625000		
dPreviousError = dTargetP - dP							0.00209600
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19790400	0.00000001	0.00390625000	0.5468750000000000000000	0.00209600
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5507812500000000000000	
call calculate()			0.19916300				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00390625000		
dPreviousError = dTargetP - dP							0.00083700
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19916300	0.00000001	0.00390625000	0.5507812500000000000000	0.00083700
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5546875000000000000000	
call calculate()			0.20042000				

calculateX: AreaUnderTDistribution: Caso normal	p = 0.2; dof = 6;						
Instruction	abs(dTargetP - dP) > dErro	dTargetP	dP	dError	dD	dX	dPreviousError
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00195312500		
dPreviousError = dTargetP - dP							-0.00042000
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20042000	0.00000001	0.00195312500	0.5546875000000000000000	-0.00042000
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5527343750000000000000	
call calculate()			0.19979200				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00097656250		
dPreviousError = dTargetP - dP							0.00020800
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19979200	0.00000001	0.00097656250	0.5527343750000000000000	0.00020800
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5537109375000000000000	
call calculate()			0.20010600				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00048828125		
dPreviousError = dTargetP - dP							-0.00010600
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20010600	0.00000001	0.00048828125	0.5537109375000000000000	-0.00010600
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5532226562500000000000	
call calculate()			0.19994900				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00024414063		
dPreviousError = dTargetP - dP							0.00005100
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19994900	0.00000001	0.00024414063	0.5532226562500000000000	0.00005100
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5534667968750000000000	
call calculate()			0.20002800				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00012207031		
dPreviousError = dTargetP - dP							-0.00002800
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20002800	0.00000001	0.00012207031	0.5534667968750000000000	-0.00002800
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5533447265625000000000	
call calculate()			0.19998800				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00006103516		
dPreviousError = dTargetP - dP							0.00001200
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19998800	0.00000001	0.00006103516	0.5533447265625000000000	0.00001200
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5534057617187500000000	
call calculate()			0.20000800				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00003051758		
dPreviousError = dTargetP - dP							-0.00000800

calculateX: AreaUnderTDistribution: Caso normal	p = 0.2; dof = 6;						
Instruction	abs(dTargetP - dP) > dErro	dTargetP	dP	dError	dD	dX	dPreviousError
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20000800	0.00000001	0.00003051758	0.5534057617187500000000	-0.00000800
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.553375244140625000000	
call calculate()			0.19999800				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00001525879		
dPreviousError = dTargetP - dP							0.00000200
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19999800	0.00000001	0.00001525879	0.5533752441406250000000	0.00000200
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.5533905029296880000000	
call calculate()			0.20000300				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00000762939		
dPreviousError = dTargetP - dP							-0.00000300
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20000300	0.00000001	0.00000762939	0.5533905029296880000000	-0.00000300
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.553382873535156000000	
call calculate()			0.20000100				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00000762939		
dPreviousError = dTargetP - dP							-0.00000100
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20000100	0.00000001	0.00000762939	0.553382873535156000000	-0.00000100
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.553375244140625000000	
call calculate()			0.19999800				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00000381470		
dPreviousError = dTargetP - dP							0.00000200
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19999800	0.00000001	0.00000381470	0.5533752441406250000000	0.00000200
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.553379058837891000000	
call calculate()			0.19999900				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00000381470		
dPreviousError = dTargetP - dP							0.00000100
while abs(dTargetP - dP) > dError	TRUE	0.2	0.19999900	0.00000001	0.00000381470	0.553379058837891000000	0.00000100
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.553382873535156000000	
call calculate()			0.20000100				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00000190735		
dPreviousError = dTargetP - dP							-0.00000100
while abs(dTargetP - dP) > dError	TRUE	0.2	0.20000100	0.00000001	0.00000190735	0.553382873535156000000	-0.00000100
dX = (dP > dTargetP) ? dX - dD : dX + dD						0.553380966186523000000	

calculateX: AreaUnderTDistribution: Caso normal	p = 0.2; dof = 6;						
Instruction	**abs(dTargetP - dP) > dErro**	**dTargetP**	**dP**	**dError**	**dD**	**dX**	**dPreviousError**
call calculate()			0.20000000				
dD = (dPreviousError * (dTargetP - dP) ? dD/2:dD					0.00000190735		
dPreviousError = dTargetP - dP							0.00000000
while abs(dTargetP - dP) > dError	FALSE	0.2	0.20000000	0.00000001	0.00000190735	0.5533809661865230000000	0.00000000

main: AreaUnderTDistribution: Caso normal 1	**p = 0.2; dof = 6;**		
Instruction	dP	iDof	dX
declare and initialise variable areXCalculator			
declare and initialise variable ioHandler			
call readValue on ioHandler with parameters: “Introduce el valor del área para la cuál se calculará x: (debe de ser numérico real, entre 0 y 0.5)”, sINVALID_PVALUE, Pattern.compile("((0+)?(\\.(([0-4]\\d*)|50*)))|((0+)(\\.(([0-4]\\d*)|50*))?)")			
parse the returned value to double			
assign the returned value to class variable dP on areXCalculator	0.2		
call readValue on ioHandler with parameters: “Introduce el valor de los grados de libertada dof: (debe de ser numérico entero y mayor a 0)”, sINVALID_INTEGER, Pattern.compile(“\\d*[1-9]\\d*")			
parse the returned value to Integer			
assign the returned value to class variable iDof on areXCalculator		6	
call the class function calculateX on areXCalculator			0.55338096
print areXCalculator	0.2	6	0.55338096

main: AreaUnderTDistribution: Caso anormal 1	**p = 0.51;**		
Instruction	dP	iDof	dX
declare and initialise variable areXCalculator			
declare and initialise variable ioHandler			
call readValue on ioHandler with parameters: “Introduce el valor del área para la cuál se calculará x: (debe de ser numérico real, entre 0 y 0.5)”, sINVALID_PVALUE, Pattern.compile("((0+)?(\\.(([0-4]\\d*)|50*)))|((0+)(\\.(([0-4]\\d*)|50*))?)")			
* Infinitely loop until the inserted dP is valid...			
* The same applies for iDof			