کلاس Genome:

این برنامه یک کلاس به نام Genome تعریف می کند که در آن متغیرهای DNA، RNA و متد هایی برای تغییر آنها و نمایش آنها تعریف شده است. برای پیاده سازی این کلاس از دو کتابخانه string و vector استفاده شده است.

متغیرهای کلاس شامل دو رشته DNA و یک رشته RNA است. دو رشته DNA به ترتیب نماینده دو رشته نوکلئوتیدی در دو شاخه مختلف DNA یک سلول هستند و رشته RNA نیز نماینده یک رشته نوکلئوتیدی است که توسط یک پروتئین ساخته شده است.

متدهای این کلاس عبارتند از:

* سازنده های کلاس که برای ایجاد شیء جدید از کلاس استفاده می شوند.
* متد createDNAFromRNA که RNA را دریافت می کند و مکمل آن را چاپ می کند.
* متد smallMutation که یک رشته و دو کاراکتر را دریافت می کند و n تای آن کاراکتر را در RNA و DNA با کاراکتر جدید جایگزین می کند.
* متد largeMutation که دو رشته ورودی دریافت می کند و هر بار که زیررشته ای با طول مشخص در RNA و DNA یافت می شود، آن زیررشته را با رشته جدید جایگزین می کند.
* متد reverse که یک زیررشته دریافت می کند و هر بار که زیررشته ای با طول مشخص در RNA و DNA یافت می شود، آن زیررشته را با نسخه معکوس آن جایگزین می کند.
* متد printRNA که RNA را چاپ می کند.
* متد printDNA که دو رشته DNA را چاپ می کند.
* متد getDNA که آرایه ای از دو رشته DNA را باز می گرداند.
* متد getComplementChar که یک کاراکتر را دریافت کرده و مکمل آن را باز می گرداند

توابعی که در این کد وجود دارند عبارتند از:

1- Genome(string rna, string dna1, string dna2): که یک constructor برای کلاس Genome است. در این constructor سه پارامتر rna، dna1 و dna2 به عنوان ورودی گرفته شده و در متغیرهای مناسب کلاس ذخیره می‌شوند.

2- Genome(string dna1, string dna2): که یک constructor برای کلاس Genome است. در این constructor دو پارامتر dna1 و dna2 به عنوان ورودی گرفته شده و در متغیرهای مناسب کلاس ذخیره می‌شوند.

3- createDNAFromRNA(): که یک متد است که DNA را از RNA ایجاد کرده و آن را چاپ می‌کند.

4- smallMutation(char find, char replace, int n): که یک متد است که n کاراکتر از RNA و DNA را با توجه به کاراکترهای find و replace جایگزین می‌کند.

5- largeMutation(string subString, string replace): که یک متد است که زیررشته‌ای به نام subString را در RNA و DNA پیدا کرده و با replace جایگزین می‌کند.

6- reverse(string subString): که یک متد است که زیررشته‌ای به نام subString را در RNA و DNA پیدا کرده و برعکس می‌کند.

7- printRNA(): که یک متد است که مقدار RNA را چاپ می‌کند.

8- printDNA(): که یک متد است که مقدار DNA را چاپ می‌کند.

9- getDNA(): که یک متد است که مقدار DNA را به صورت یک اشاره‌گر به آرایه‌ای از رشته‌ها بازمی‌گرداند.

همچنین این کد از دو کتابخانه <string> و <vector> استفاده می‌کند و از namespace std نیز استفاده شده است.

توضیح خط به خط smallMutation

این تابع یک جینوم را به عنوان ورودی دریافت می‌کند و نوعی از جهش ژنتیکی به نام جهش کوچک (smallMutation) را اعمال می‌کند.

ورودی‌های تابع:

* find: یک کاراکتر که برای جستجوی آن در جینوم استفاده می‌شود.
* replace: یک کاراکتر که به جای کاراکتر‌های یافت شده در جینوم جایگزین می‌شود.
* n: تعداد جایگزینی که باید انجام شود.

توضیح خط به خط کد:

1. در این خط، ابتدا مقدار ورودی n در متغیر موقت temp ذخیره می‌شود.
2. در این خط شرطی چک می‌شود که در صورتی که طول RNA بیشتر از صفر باشد و همچنین i کمتر از طول RNA و temp بزرگتر از صفر باشد، داخل حلقه وارد می‌شود.
3. در این خط، شرطی چک می‌شود که در صورتی که کاراکتر i ام در جینوم RNA با کاراکتر find برابر باشد، مقدار temp کاهش داده شده و کاراکتر i ام جینوم RNA با کاراکتر replace جایگزین می‌شود.
4. در این خط، مقدار temp به مقدار ورودی n برگردانده می‌شود.
5. در این خط، حلقه جدیدی شروع می‌شود که برای حلقه اول، طول DNA[0] و برای حلقه دوم، طول DNA[1] در نظر گرفته شده است.
6. در این خط، شرطی چک می‌شود که در صورتی که کاراکتر i ام در DNA[0] با کاراکتر find برابر باشد، مقدار temp کاهش داده شده و همچنین کاراکتر i ام در DNA[0] با کاراکتر replace جایگزین می‌شود. همچنین، کاراکتر متناظر در DNA[1] نیز با استفاده از تابع getComplementChar که نوعی تابع کمکی است، جایگزین می‌شود.

۷. پس از اعمال تغییرات مورد نظر، آیا ژنوم به درستی به روز شده است؟

پس از اعمال تغییرات مورد نظر، ژنوم به درستی به روز شده است زیرا در هر مرحله از کد، ما مقدار رشته را مستقیماً تغییر می دهیم. همچنین، برای تغییر DNA ، ما مقدار رشته را مستقیماً تغییر داده و دیگر نیازی به بازسازی رشته نیست.

۸. در این کد از چه روشی برای تغییر مقدار یک عنصر در رشته استفاده شده است؟

در این کد، از روش مستقیم تغییر مقدار یک عنصر در رشته با استفاده از نشانگر مقدار (indexing) استفاده شده است. به عنوان مثال، برای تغییر مقدار RNA[i] ، ما به ازای هر عنصر در رشته RNA ، محتوای آن را بررسی می کنیم و در صورتی که با مقدار find مطابقت داشت، با استفاده از نشانگر i مقدار را با replace جایگزین می کنیم.

۹. چرا در دومین حلقه از اندیس ۰ برای شروع حلقه استفاده شده است؟

در دومین حلقه از اندیس ۰ برای شروع حلقه استفاده شده است زیرا برای تغییر DNA ، ابتدا باید همه عناصر DNA[0] و DNA[1] با هم متناظر شوند. در واقع، هر عنصر در DNA[0] و DNA[1] با یک عنصر متناظر در DNA[1] و DNA[0] تعیین می شود. با توجه به اینکه هر سه رشته همان طول را دارند، ما از اندیس ۰ برای شروع حلقه استفاده می کنیم.

توضیح خط به خط largeMutation

این تابع یک جایگزینی بزرگ در ژنوم انجام می‌دهد. برای انجام این کار، ابتدا برای رشته RNA ژنوم ورودی، به طول زیررشته مشخص شده جستجو می‌شود و اگر زیررشته مشخص شده پیدا شود، با رشته جدیدی که به عنوان جایگزین مشخص شده است، جایگزین می‌شود. سپس برای دو رشته DNA ژنوم ورودی، همان عملیات برای هر دو رشته به صورت جداگانه انجام می‌شود.

این تابع دو رشته subString و replace را به عنوان ورودی دریافت می کند و در صورت وجود زیررشته subString در RNA یا DNA ژنوم، آن را با رشته replace جایگزین می کند.

خطوط کد تابع به شرح زیر است:

1. برای جستجو در RNA از شمارنده i برای حرکت در طول رشته RNA استفاده می شود. در صورتی که طول رشته subString بیشتر از طول رشته RNA باشد یا رشته subString در RNA وجود نداشته باشد، حلقه تمام می شود.
2. اگر رشته subString در RNA وجود داشته باشد، آن رشته با رشته replace جایگزین می شود و حلقه پایان می یابد.
3. در حلقه دوم، از شمارنده i برای جستجو در طول رشته های DNA[0] و DNA[1] استفاده می شود. در صورتی که طول رشته subString بیشتر از طول هر یک از این دو رشته باشد یا subString در هیچ کدام از این دو رشته وجود نداشته باشد، حلقه پایان می یابد.
4. اگر subString در رشته DNA[0] وجود داشته باشد، subString با رشته replace جایگزین می شود و مکمل replace به عنوان رشته جایگزین در رشته DNA[1] استفاده می شود. حلقه پایان می یابد.
5. در غیر این صورت، اگر subString در رشته DNA[1] وجود داشته باشد، subString با replace جایگزین می شود و مکمل replace به عنوان رشته جایگزین در رشته DNA[0] استفاده می شود. حلقه پایان می یابد.

توضیح خط به خط reverse

این متد یک رشته زیر رشته را به صورت معکوس جایگزین می‌کند. توضیحات خط به خط آن به شرح زیر است:

* در این خط، یک متغیر برای نگهداری رشته معکوس شده ایجاد شده است که مقدار آن با فراخوانی تابع reverseString با استفاده از رشته subString تعیین می‌شود.
* در این حلقه، رشته RNA بررسی می‌شود و اگر زیر رشته subString در آن یافت شد، با استفاده از تابع replace آن زیررشته با معکوس شده‌ی آن جایگزین می‌شود.
* در این حلقه نیز، رشته DNA[0] بررسی می‌شود. اگر زیر رشته subString در آن یافت شد، با استفاده از تابع replace آن زیررشته با معکوس شده‌ی آن جایگزین می‌شود و همچنین در رشته DNA[1] نیز زیررشته معکوس شده‌ی آن جایگزین می‌شود.
* اگر زیر رشته subString در رشته DNA[0] یافت نشد، در این حلقه بررسی می‌شود که آیا زیررشته subString در رشته DNA[1] یافت می‌شود یا نه. اگر یافت شد، با استفاده از تابع replace آن زیررشته با معکوس شده‌ی آن جایگزین می‌شود و همچنین در رشته DNA[0] نیز زیررشته معکوس شده‌ی آن جایگزین می‌شود.

کلاس Cell:

این کلاس یک شخصیت سلول را نمایش می دهد که از ژنوم تشکیل شده است. در این کلاس متدهایی برای انجام عملیات مختلف بر روی ژنوم های سلول موجود است، از جمله مرگ سلول به علت عدم صحت ژنوم، جایگزینی یک حرف با یک حرف دیگر، جایگزینی یک رشته با رشته دیگر، تغییر ژنوم بزرگ و جستجوی پالیندروم های کامل در ژنوم ها. همچنین برخی از متدهای دیگر نیز برای برگشت اطلاعات ویژگی های سلولی مانند تعداد کروموزوم ها و ژنوم های موجود در سلول ارائه شده است.

این برنامه یک شبیه‌ساز سلول است که شامل کلاس‌های "سلول" و "ژنوم" است. هر سلول شامل یک تعدادی کروموزوم است که هر کروموزوم از دو رشته یکسان تشکیل شده است. کلاس ژنوم شامل توابعی است که برای اعمال جهش‌های کوچک و بزرگ به رشته ژنتیکی استفاده می‌شود و همچنین برای پیدا کردن زیررشته‌های پالیندروم و جستجو برای نقص در کروموزوم کاربرد دارد.

در کلاس سلول، توابعی برای مرگ، جهش، برعکس کردن و پیدا کردن زیررشته‌های پالیندروم تعریف شده‌اند. همچنین توابعی برای چاپ کروموزوم‌ها و بررسی صحت آن‌ها نیز وجود دارد. در نهایت، توابعی برای بازگرداندن تعداد کروموزوم‌ها و کروموزوم‌های خود سلول تعریف شده‌اند.

متد death

متد death یک حلقه دارد که تمامی کروموزوم های سلول را چک می‌کند. اگر یک کروموزوم نامعتبر باشد، پیغامی چاپ می‌شود که می‌گوید سلول به دلیل ناهنجاری‌های کروموزومی مرده است.

متد isValidChromosome برای چک کردن صحت یک کروموزوم استفاده می‌شود. این متد یک جمله‌ی بازگشتی دارد که یک رشته از کروموزوم را به عنوان ورودی می‌گیرد. این رشته، دو رشته دیگر (قسمت‌های توأم) که هر کدام حاوی مجموعه‌ای از نوکلئوتیدها هستند را تشکیل می‌دهد. سپس این متد هر نوکلئوتید را با استفاده از رشته تکمیل‌کننده آن در کروموزوم دوم چک می‌کند. اگر تعداد نوکلئوتیدهای نامعتبر بیشتر از ۵ باشد، این متد false برمی‌گرداند.

متد largeMutation

این متد یک سلول را در نظر می گیرد و دو رشته DNA را به عنوان ورودی می گیرد. سپس با استفاده از شماره هر کدام از دو کروموزوم مورد نظر، رشته های DNA آن کروموزوم را از روی مجموعه کروموزوم های سلول دریافت می کند.

سپس این متد به دنبال دو موقعیت آغازین این دو رشته در دو کروموزوم مربوطه می گردد و اگر چنین موقعیتی وجود نداشته باشد، یک پیام خطا چاپ می کند و از ادامه اجرای متد منصرف می شود.

در صورتی که موقعیت مورد نظر پیدا شد، با استفاده از تابع "largeMutation" که برای هر کروموزوم قابل دسترسی است، دو رشته را جایگزینی می کند. سپس، کروموزوم های مورد نظر را چاپ کرده و تغییرات را نشان می دهد.

متد smallMutation

این متد یک مولکول DNA را در کروموزوم m با جایگزینی یک کاراکتر با یک کاراکتر دیگر تغییر می دهد.

* ابتدا یک کاراکتر جستجویی و کاراکتر جایگزین را به عنوان ورودی دریافت می کند.
* سپس رشته DNA را از کروموزوم m دریافت می کند و موقعیت کاراکتر جستجویی را در کل رشته DNA پیدا می کند.
* در صورتی که کاراکتر مورد نظر در رشته DNA یافت نشود، یک پیام خطا چاپ کرده و متد را ترک می کند.
* در غیر این صورت، کروموزوم m تغییر می کند. یک نسخه جدید از DNA ایجاد می شود که در آن کاراکتر جستجویی با کاراکتر جایگزین جایگزین شده است و این نسخه به کروموزوم m نسبت داده می شود.
* در نهایت، DNA کروموزوم m پس از اعمال تغییرات، چاپ می شود.

متد reverse

این متد برای برعکس کردن یک رشته در کروموزوم مشخص در سلول استفاده می‌شود.

خط ۲: پیامی در خروجی چاپ می‌شود که حاوی دو مقدار s1 و n است و نشان دهنده ورودی‌های این تابع است.

خط ۴: با فراخوانی تابع printDNA() مقدار کروموزوم مورد نظر چاپ می‌شود.

خط ۶: با فراخوانی تابع getDNA() از کلاس کروموزوم، دو رشته DNA[0] و DNA[1] دریافت می‌شود.

خط ۷: با استفاده از ترکیب دو رشته DNA[0] و DNA[1] و جستجوی s1 در آن، اولین موقعیت s1 در کروموزوم مورد نظر (chromosomes[n]) پیدا می‌شود و در متغیر pos ذخیره می‌شود.

خط ۸: اگر مقدار pos برابر با یک عدد منفی باشد، به این معنی است که رشته مورد نظر در کروموزوم موجود نبوده و پیام "Substring not found in n" چاپ می‌شود.

خط ۱۰: با فراخوانی تابع reverse(s1) رشته s1 در کروموزوم n برعکس می‌شود.

خط ۱۲: پس از انجام عملیات reverse، با فراخوانی تابع printDNA() مقدار کروموزوم مورد نظر چاپ می‌شود.

متد findComplementPalindromes

این برنامه یک شیء از کلاس Cell را به عنوان ورودی می‌گیرد. ابتدا برای هر کروموزوم در شیء Cell، ژنوم آن را با گرفتن DNA آن و اعمال دو حلقه for برای پیدا کردن زیررشته‌های پالیندرمیک تحلیل شده و در صورتی که یک زیررشته پالیندرمیک پیدا شود، رشته‌ی کامل و نیز کامل کنشگر نظیر آن با استفاده از دو تابع isPalindrome و reverseComplement پرینت می‌شود.

* تابع isPalindrome: ورودی یک رشته‌ی str است که بررسی می‌کند که آیا این رشته پالیندرم است یا نه. ابتدا طول رشته را بدست آورده و سپس با یک حلقه for برای نیمی از رشته، هر کاراکتر را با معکوس آن در نیمی دیگر رشته مقایسه می‌کند. اگر همه کاراکترهای رشته با معکوس آنان برابر باشند، تابع true را برمی‌گرداند.
* تابع reverseComplement: ورودی یک رشته‌ی str است که بازگشت رشته‌ی معکوس و نیز تبدیل هر کاراکتر به نظیر کامل کنشگرش (مانند A به T و T به A و ...) را با یک حلقه for انجام می‌دهد. در نهایت، رشته‌ی نهایی به عنوان خروجی برگشت داده می‌شود.

در تابع اصلی، برای هر کروموزوم در شیء Cell، با گرفتن DNA آن و اعمال دو حلقه for برای پیدا کردن زیررشته‌های پالیندرمیک، زیررشته‌های پالیندرمیک و کامل کنشگر آن با استفاده از توابع isPalindrome و reverseComplement پرینت می‌شود.

کلاس Animal

این کلاس یک شیء حیوان را مدلسازی می کند که شامل یک سری سلول است که هر یک از آنها شامل ژنومی است. این کلاس متدهایی برای محاسبه شباهت ژنتیکی با حیوانات دیگر، تولید مثل بدون جنسیت و تولید مثل جنسی دارد. همچنین این کلاس متد cellularDeath را برای مرگ سلولی تمامی سلول های حیوان دارد.

این کلاس برای مدل سازی حیوانات استفاده می شود. دو روش تولید جدید با روش افزایشی و جنسی ارائه شده است. همچنین روش هایی برای محاسبه شباهت ژنتیکی بین حیوانات و تلف شدن سلولی هم در این کلاس تعریف شده است.

برای هر حیوان که از یک سری سلول تشکیل شده است، می توان تابع calculateGeneticSimilarity را فراخوانی کرد که این تابع یک حیوان دیگر را به عنوان ورودی دریافت می کند و شباهت ژنتیکی آنها را محاسبه می کند.

تابع asexualReproduction یک تولید جدید از حیوان را تولید می کند که با روش افزایشی است. برای این کار یک تعدادی سلول تولید شده و سلول هایی که در اندازه نمی گنجند، تکمیل می شوند.

عملیات تلف شدن سلولی نیز با تابع cellularDeath انجام می شود که همه سلول ها را تلف می کند.

در روش جنسی، ابتدا بررسی می شود که حیواناتی که می خواهند تولید کنند تعداد زوج کروموزوم داشته باشند. سپس نیمی از کروموزوم ها را از هر دو حیوان به صورت تصادفی انتخاب کرده و یک حیوان جدید با آنها تولید می کند.

متد calculateGeneticSimilarity

این تابع شباهت ژنتیکی بین دو حیوان Animal و otherAnimal را با استفاده از شمارش تعداد ژن‌های مشترک در هر سلول، محاسبه می‌کند. ابتدا یک متغیر برای ذخیره مقدار شباهت ایجاد شده و برابر با صفر قرار می‌دهد. سپس برای هر سلول، تعداد ژنوم‌های مشترک با سلول دیگر محاسبه شده و در متغیر commonGenes قرار می‌گیرد. در انتها، شباهت ژنتیکی سلول‌های حیوان با همدیگر به صورت میانگین در متغیر similarity ذخیره شده و با توجه به تعداد سلول‌ها و درصد ژنتیکی محاسبه شده، برای هر دو حیوان مقدار بازگشتی محاسبه می‌شود.

ین تابع یک جانور دیگر را به عنوان پارامتر دریافت می‌کند و شباهت ژنتیکی بین این دو جانور را محاسبه می‌کند. برای این کار، برای هر سلول در هر جانور، تعداد ژن‌های مشترک با سلول معادل در جانور دیگر را محاسبه می‌کند و به تعداد کل ژن‌های آن سلول تقسیم می‌کند. سپس، میانگین شباهت ژنتیکی را برای همه سلول‌ها در جانور محاسبه می‌کند و به صد درصد تبدیل می‌کند و در نهایت این مقدار به عنوان خروجی تابع بازگردانده می‌شود.

در مرحله اول، یک متغیر double با نام similarity تعریف می‌شود و مقدار اولیه آن برابر با صفر است. همچنین تعداد سلول‌های جانور فعلی در متغیر cellCount ذخیره می‌شود.

در مرحله بعدی، یک حلقه for برای هر سلول در جانور فعلی اجرا می‌شود. در هر دور این حلقه، یک متغیر int با نام commonGenes تعریف می‌شود و مقدار اولیه آن برابر با صفر است. همچنین تعداد کروموزوم‌های هر سلول در جانور فعلی در متغیر geneCount ذخیره می‌شود. سپس یک حلقه for دیگر برای هر ژن در هر کروموزوم در سلول فعلی اجرا می‌شود. در هر دور این حلقه، با استفاده از شرط درون if، بررسی می‌شود که آیا همین ژن در سلول معادل در جانور دیگر وجود دارد یا خیر؟ اگر وجود داشت، مقدار متغیر commonGenes یک واحد افزایش پیدا می‌کند.

در مرحله بعدی، مقدار commonGenes تقسیم بر تعداد کل ژن‌های سلول فعلی در متغیر similarity ذخیره می‌

متد asexualReproduction

این متد برای تولید جنسی بدون جنسیت (asexual reproduction) استفاده می‌شود. در این روش، یک فرد بدون نیاز به همجنس خود می‌تواند تکثیر شود.

اولین کاری که این متد انجام می‌دهد، شمارش تعداد سلول‌ها در این حیوان (animal) است. سپس یک وکتور جدید (new\_cells) ساخته می‌شود. سلول‌های قدیمی که شانس بیشتری برای به وجود آوردن نسل بعدی دارند، در این وکتور جدید قرار می‌گیرند.

این انتخاب با استفاده از باقی‌مانده تعداد سلول‌ها (remainedSize) انجام می‌شود. در هر گام، اگر اندیس (i) برای سلول فعلی بر ۷ بخش‌پذیر باشد، به وکتور جدید اضافه می‌شود. در غیر این صورت، سلول جدید با یک کروموزوم (chromosome) خالی ساخته شده و به وکتور اضافه می‌شود.

در نهایت، یک حیوان جدید با وکتور سلول‌های جدید ساخته می‌شود و به عنوان نتیجه برگردانده می‌شود.

این متد برای شبیه‌سازی فرآیند تولید مثل بدون تلقیح است. در این فرآیند، پدر و مادر به صورت جداگانه و مستقل از هم به تولید جوجه می‌پردازند.

در این متد، ابتدا تعداد سلول‌های حیوان را در متغیر num\_cells ذخیره می‌کنیم. سپس یک بردار از سلول‌های جدید با نام new\_cells تعریف می‌کنیم.

سپس، در یک حلقه for، برای هر یک از سلول‌های اولیه، چک می‌کنیم که آیا اندیس آن به ازای باقی‌مانده تقسیم بر ۷ برابر با صفر است یا نه. اگر برابر با صفر باشد، آن را به new\_cells اضافه می‌کنیم. در واقع، این بدان معنی است که سلول جدیدی با همان مشخصات سلول اولیه تولید شده است.

در ادامه، مقدار remainedSize محاسبه می‌شود که برابر است با تعداد کل سلول‌ها منهای تعداد سلول‌هایی که در مرحله قبل به new\_cells اضافه شده‌اند.

سپس، در یک حلقه دیگر، به تعداد remainedSize، یک سلول جدید با همان مشخصاتی که در متد سازنده Cell با شماره یک و بدون کروموزوم ایجاد می‌شود، به new\_cells اضافه می‌شود.

در نهایت، یک حیوان جدید با استفاده از بردار new\_cells ایجاد می‌شود و بازگشت داده می‌شود.

عملگر operator+

ین تابع، یک جفت حیوان را به عنوان ورودی دریافت می‌کند و با همان یک جفت حیوان، حیوان جدیدی را ایجاد می‌کند که کروموزوم‌های آن از نیمی از کروموزوم‌های هر یک از حیوان‌ها انتخاب شده‌اند.

با اولین شرط در این تابع، اطمینان حاصل می‌شود که هر دو حیوان در تعداد کروموزوم‌ها فرد ندارند، در غیر این صورت خطا می‌دهد.

در قسمت بعدی، برای هر کروموزوم حیوان جدید، نیمی از آن از حیوان اول و نیمی دیگر آن از حیوان دوم انتخاب می‌شود. سپس، حیوان جدید با استفاده از کروموزوم‌های ایجاد شده، ساخته می‌شود و به عنوان خروجی برگردانده می‌شود.

این تابع متد جمع دو حیوان برای تولید نسل جدید با داشتن ویژگی‌های هر دو حیوان است. قبل از شروع عمل جمع، یک شرط بررسی می‌شود تا مطمئن شدن از اینکه هر دو حیوان تعداد زوجی کروموزوم دارند. اگر یکی از آن‌ها تعدادی کروموزوم فرد داشته باشد، پیام خطا چاپ شده و یک حیوان خالی به عنوان خروجی تابع برگردانده می‌شود.

در صورتی که هر دو حیوان تعداد زوجی کروموزوم داشته باشند، به ایجاد نسل جدید می‌پردازیم. ابتدا یک بردار خالی جهت نگهداری کروموزوم‌های جدید ایجاد می‌شود. سپس تعداد کروموزوم‌های هر حیوان بدست می‌آید و با حلقه‌ای که برای همه کروموزوم‌ها اجرا می‌شود، نیمی از کروموزوم‌ها به صورت تصادفی از هر حیوان برای ایجاد نسل جدید انتخاب می‌شوند و به بردار جدید اضافه می‌شوند. سپس حیوان جدید با استفاده از کروموزوم‌های جدید ایجاد می‌شود و به عنوان خروجی تابع بازگردانده می‌شود.