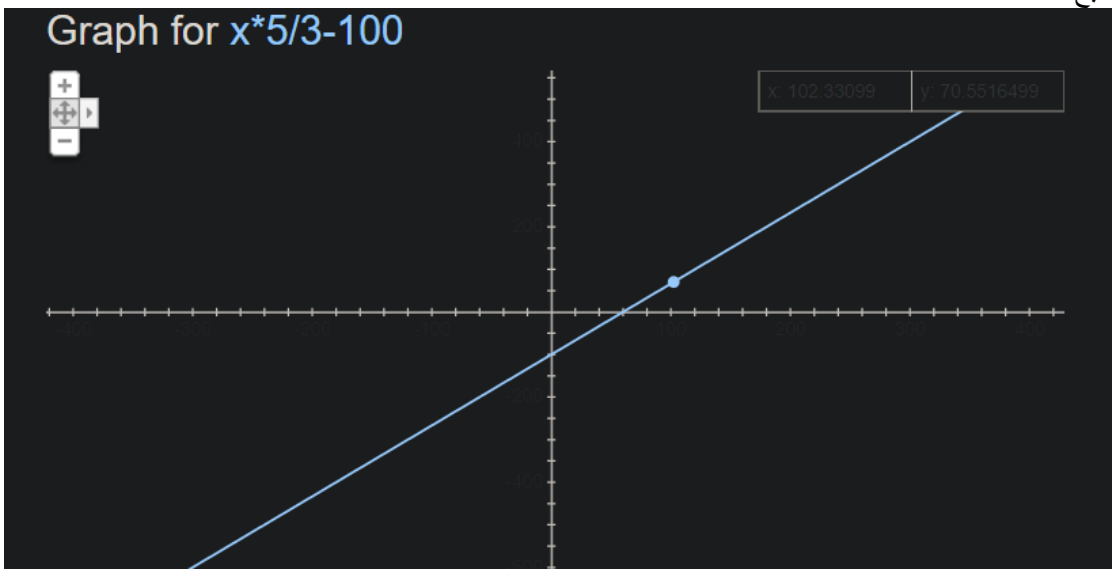
به نام خدا

آزمایش اول:

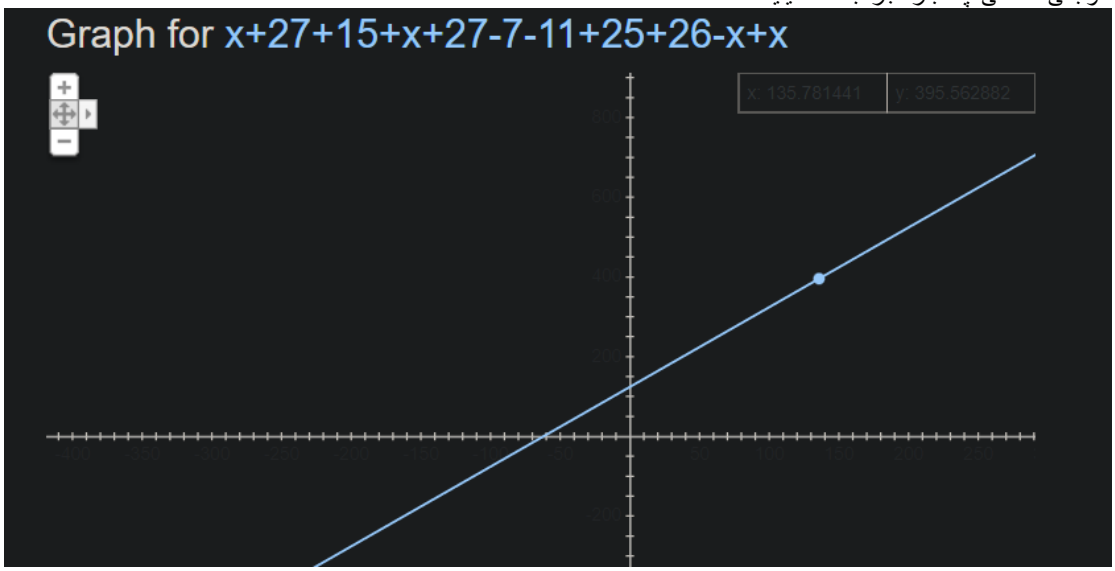
عملگر های موجود = (+) و (-)

```
2 references | 1 reference | 1 reference | 1 reference
const int limit_of_operator = 2 , limit_for_depth =10 , limit_for_constant = 100 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 2 , maximum_variable_value = 5000 , minumum_variable_value = -5000 ;
10 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 50, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
```

تابع هدف

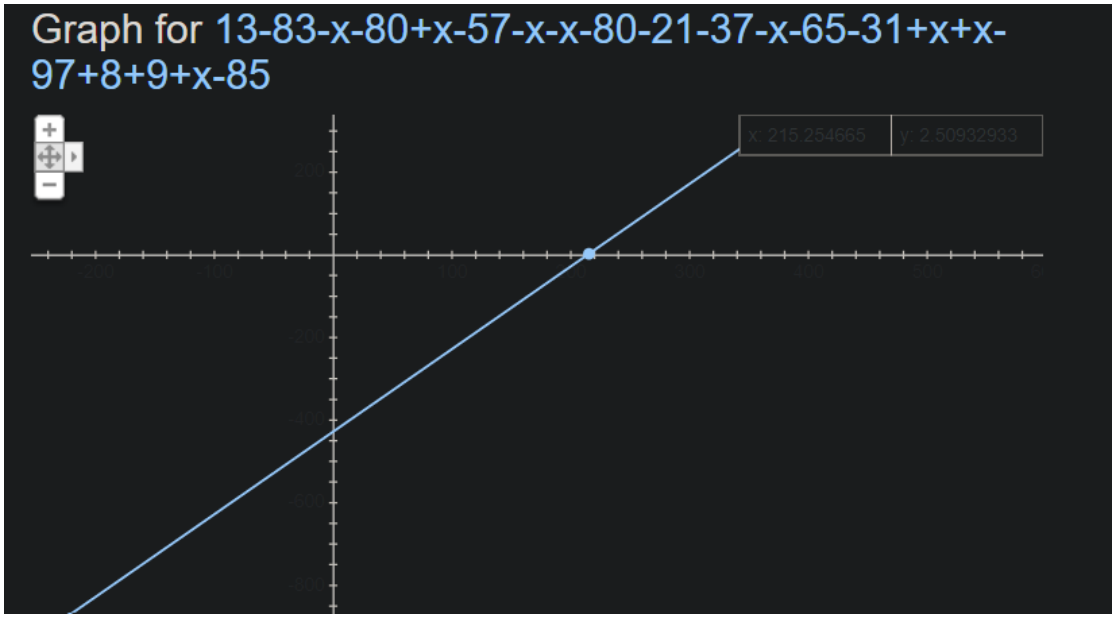


توابعی که طی چند بار اجرا بدست میاید

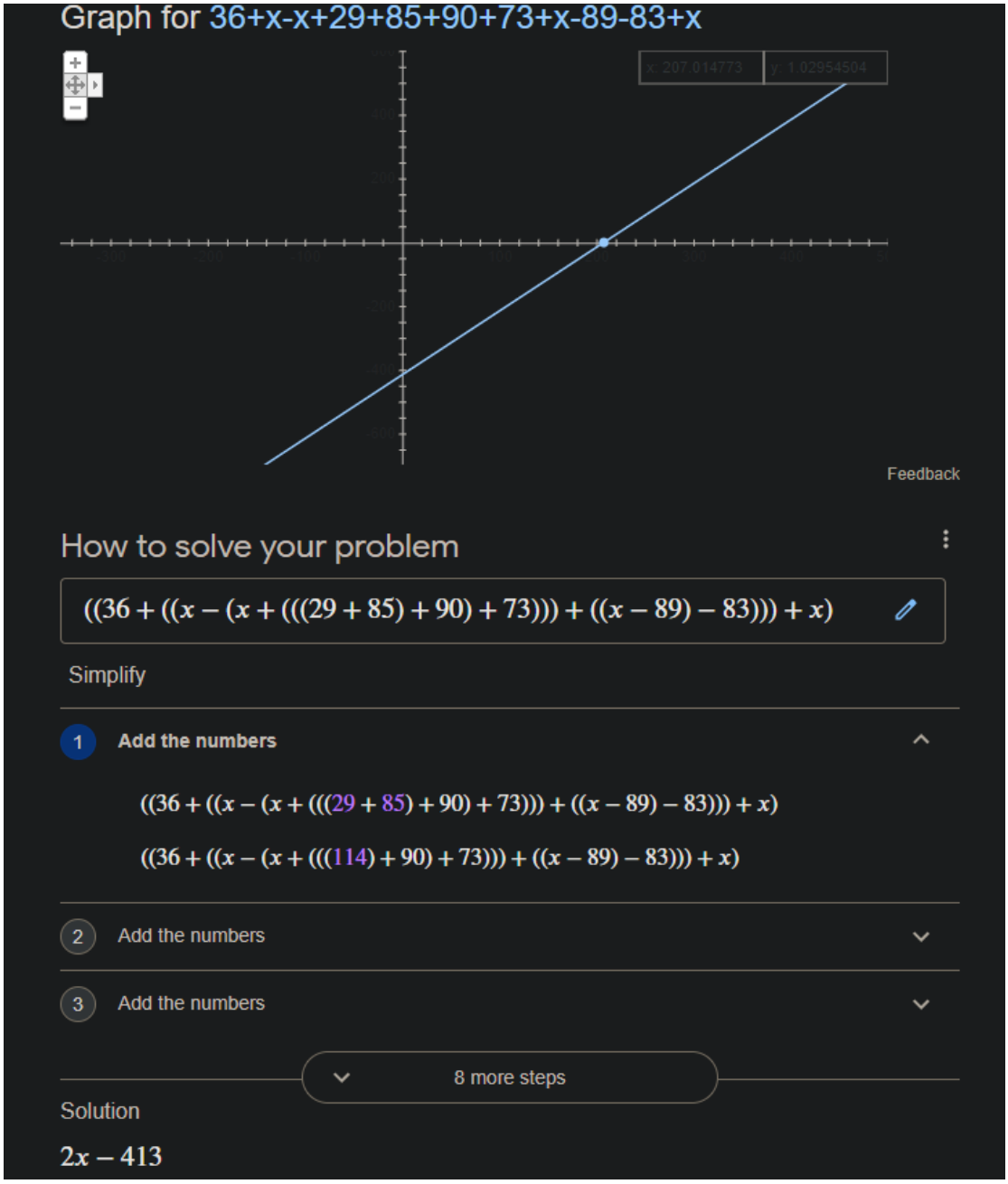


و

Mse:535237


Graph for 13-83-x-80+x-57-x-x-80-21-37-x-65-31+x+x-97+8+9+x-85
x: 215.254665
y: 2.50932933

Mse:343529

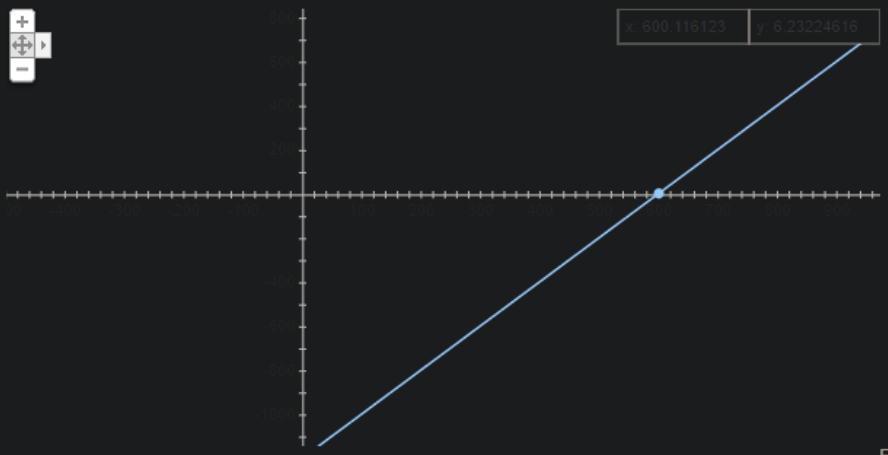


با تغیر تنظیمات اولیه دوباره مقادیر را حساب میکنیم

```
2 references | 1 reference | 1 reference | 1 reference
const int limit_of_operator = 2 , limit_for_depth =20 , limit_for_constant = 500 , limit_for_type = 3
4 references | 2 references | 2 references
const int test_case = 2 , maximum_variable_value = 5000 , minumum_variable_value = -5000 ;
10 references
static  int tree_instance_count = 1000 ;
1 reference | 1 reference
const int last_generation_percent = 50, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
```

Mse:254020

Graph for x-49+x-x+x-180+x-x+x+53-60+463+x-x+x+387+108



Feedback

How to solve your problem

$$(((((x-49)+(x-x))+(x-180))+(((x-x)+x)+53))-(60+((((...$$

Grouping

1 Combine like terms

$$(((((x-49)+(0))+(x-180))+(((x-x)+x)+53))-(60+((((463+x)-x)+...$$

2 Add the numbers

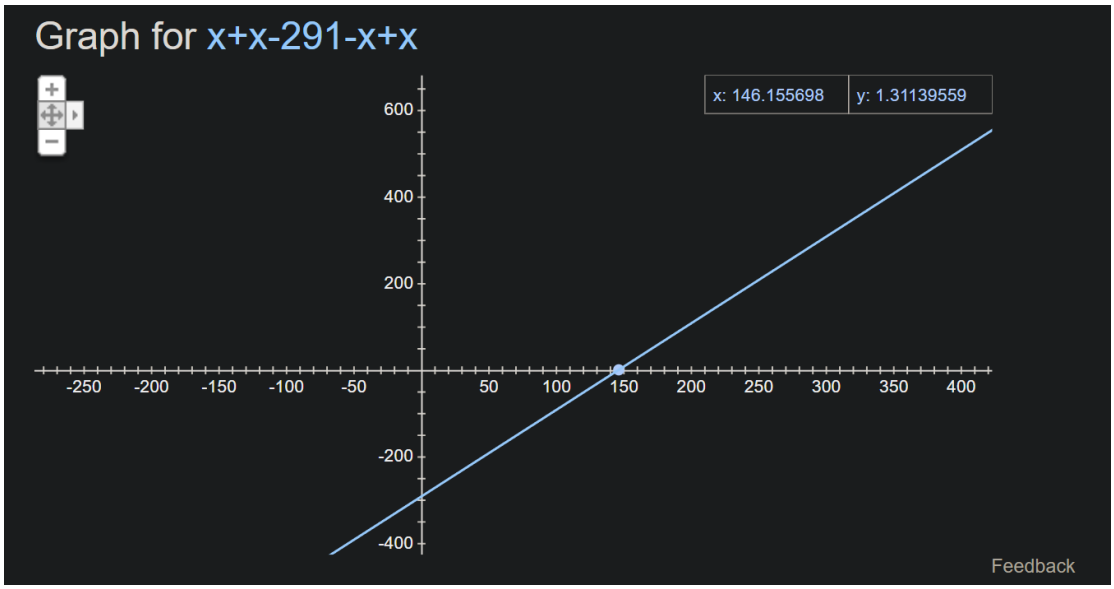
3 Subtract the numbers

16 more steps

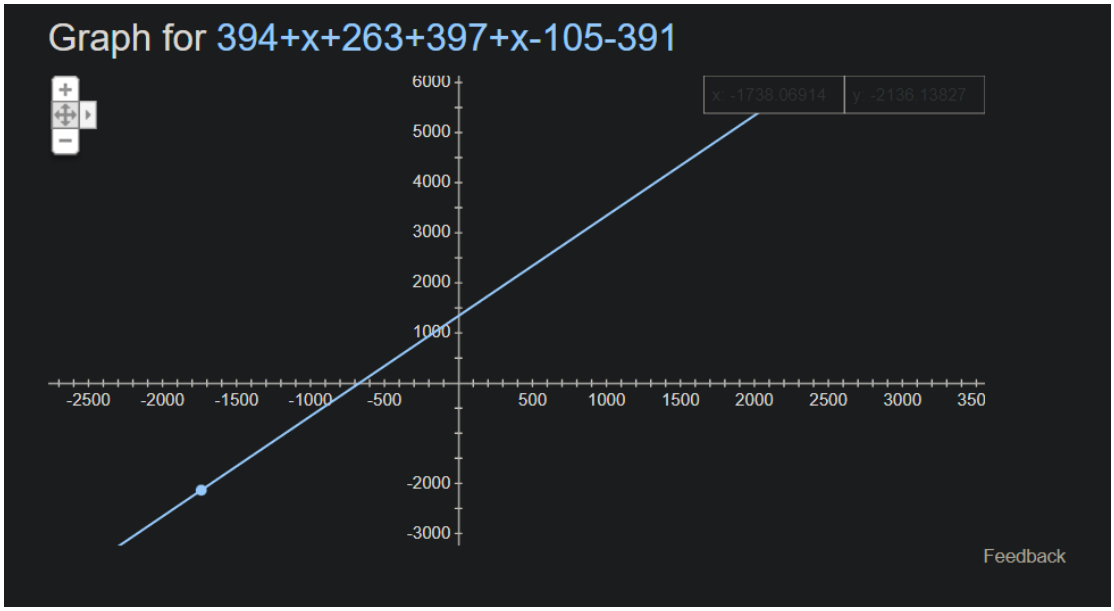
Solution

$$2(x-597)$$

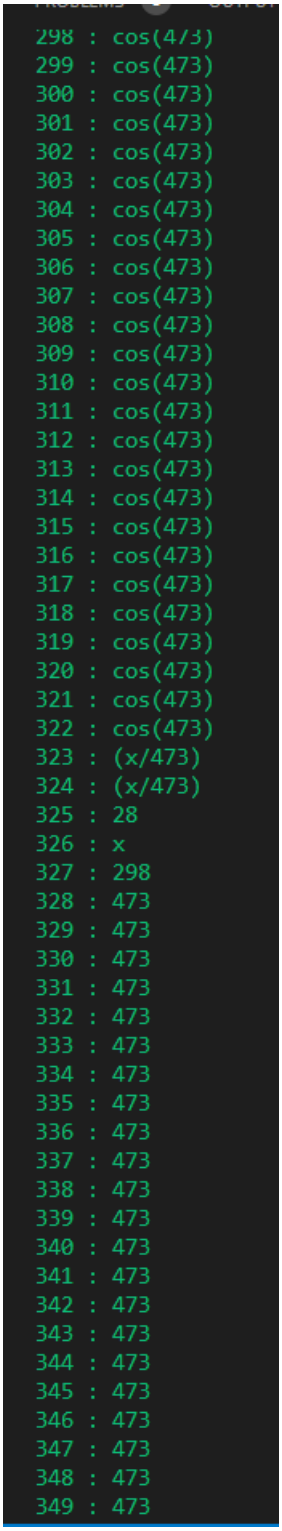
Mse:144020



Mse:31152



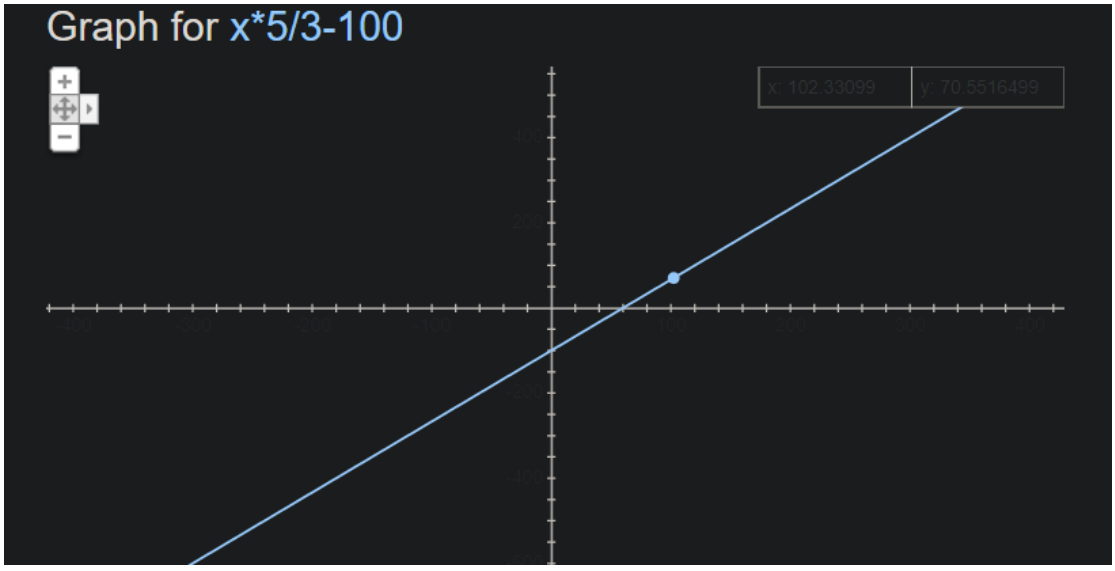
حال تابع را با اپراتور های مختلف تجهیز کردم
تست کیس ها همچنان به صورت اینت هستند
تعداد تست کیس ها کم هست و به صورت رندوم فقط اول برنامه یک با ر ایجاد میشوند چون هرچی بیشتر میریم جلو معلوم میشه که نقاط بهتر میشه یا نه
مشکلی که باهاش روبرو شدم این بود که بعد از چند بار تولید مثل تعداد درخت های مثل هم بسیار زیاد میشد



برای حل این مشکل اگر درخت های پشت سر هم با هم یکی بودند ، به جاش یه درخت جدید قرار میدهیم

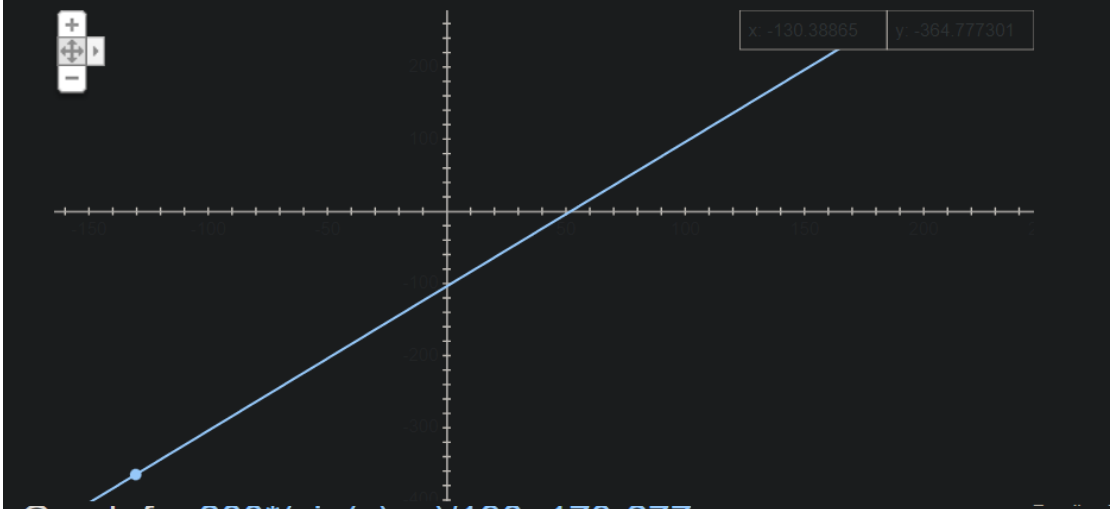
تابع شایستگی:با توجه به فرمول ام اس ای بدست اوردم
تولید جمعیت اولیه:به صورت کاملا رندوم و ارتفاع درخت هم به صورت رندوم
نحوه انتخاب والدین:ابتدا با استفاده از درصد های اولیه ، نسل خوب را با هم کراس اور و سپس نسل خوب را با نسل بد کراس اور میکنیم و
نحوه تولید مثل:همان موقع که نسل جدید ایجاد میشود با یکی از درخت های موجود جایگزین میکنیم
نحوه ترکیب متقاطع و جهش:هنگام انتخاب نود برای ترکیب کردن ، با احتمال کمی ممکن هست که هویت هر یک گره ها ازعوض شود
شرط خاتمه:هزار بار تولید مثل انجام میشود
چالش ها :طی گزارش نوشته شده

Target function:

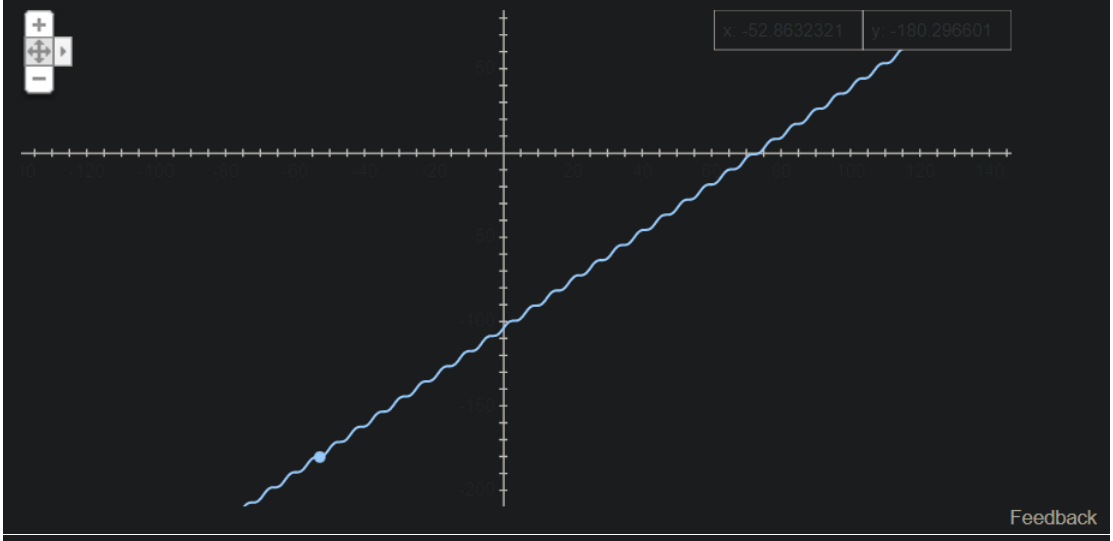


Result function:

Graph for x-104+x

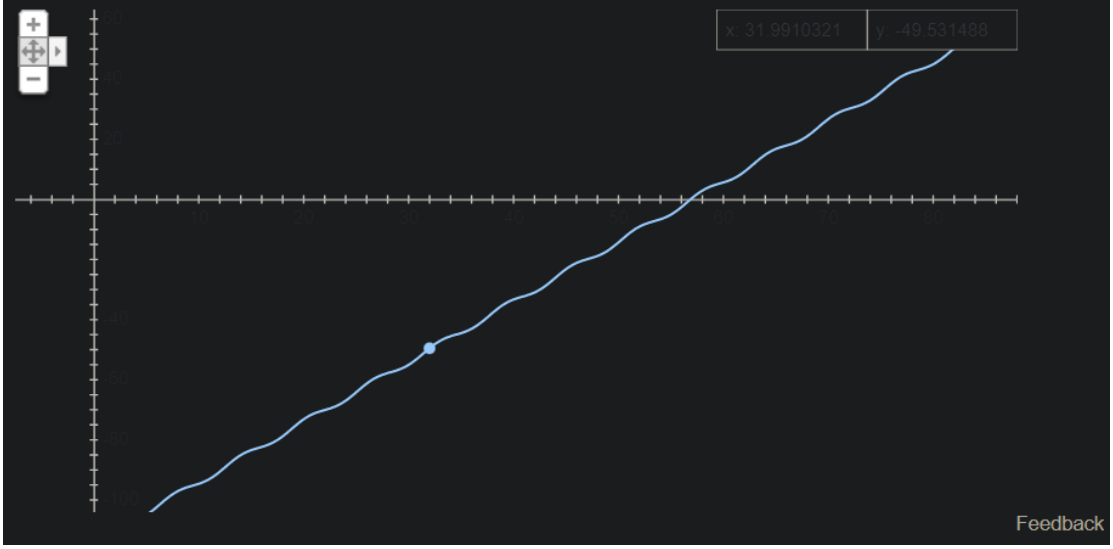


Graph for 283*(sin(x)+x)/198+173-277



Feedback

Graph for x-115-x-sin(218)-sin(x)

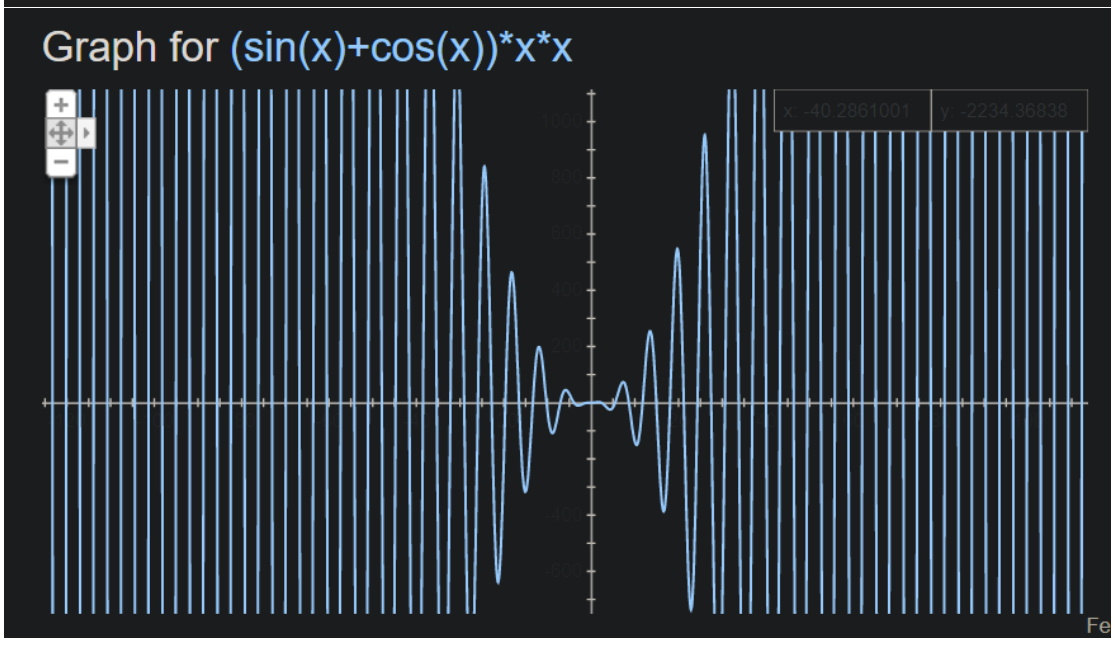


Feedback

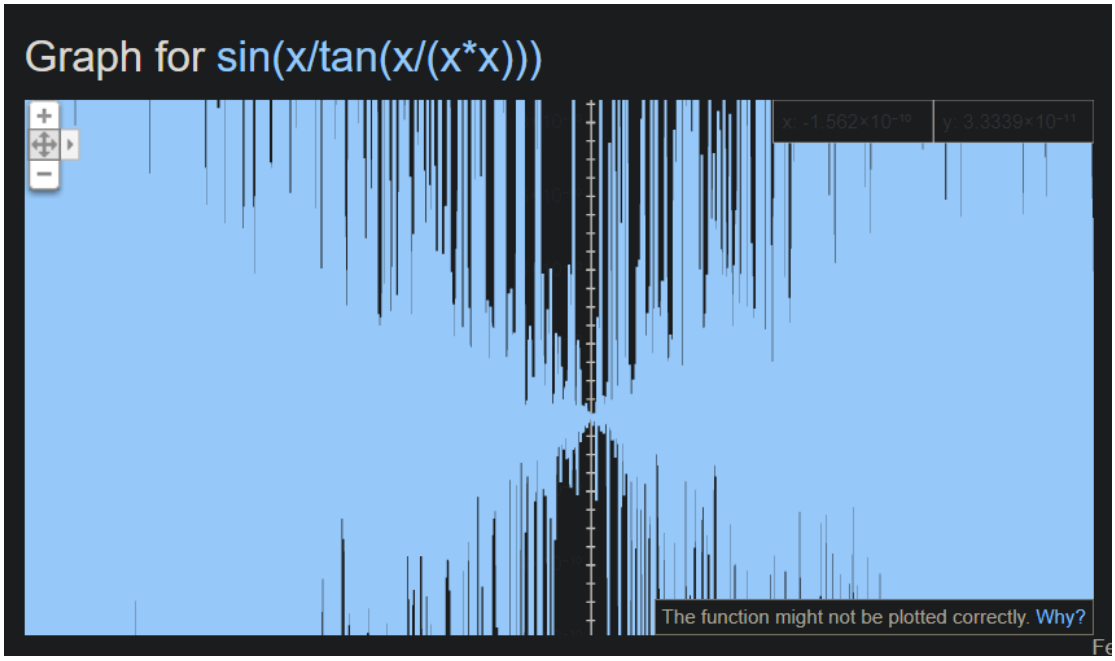
```
trees[0] with mse :1455.8406491008418
(((283*(sin(x)+x))/198)+(173-277))
PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :3530.541911027924
(x-(115-(x-(sin(218)-sin(x)))))
PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :2120.8
((x-88)+x)
```

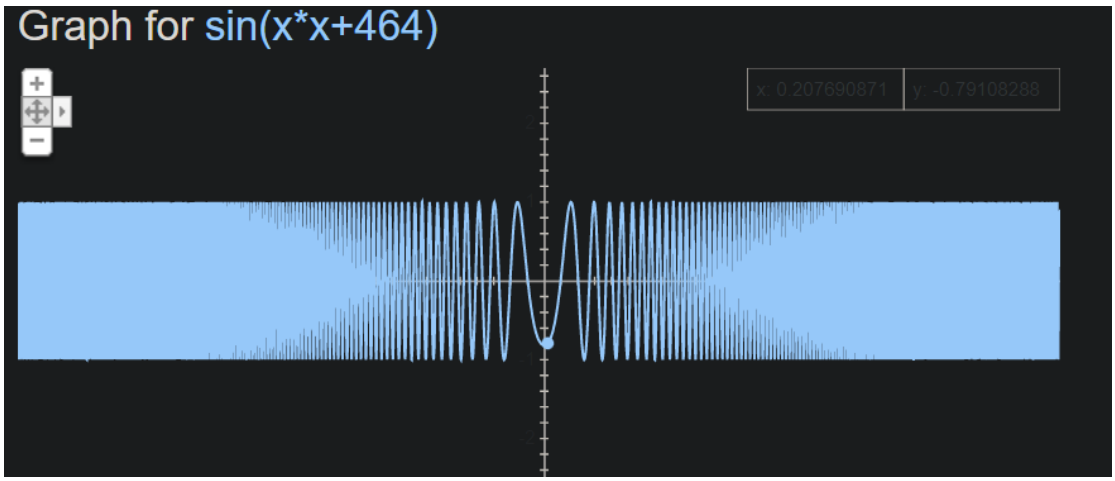
Target function:

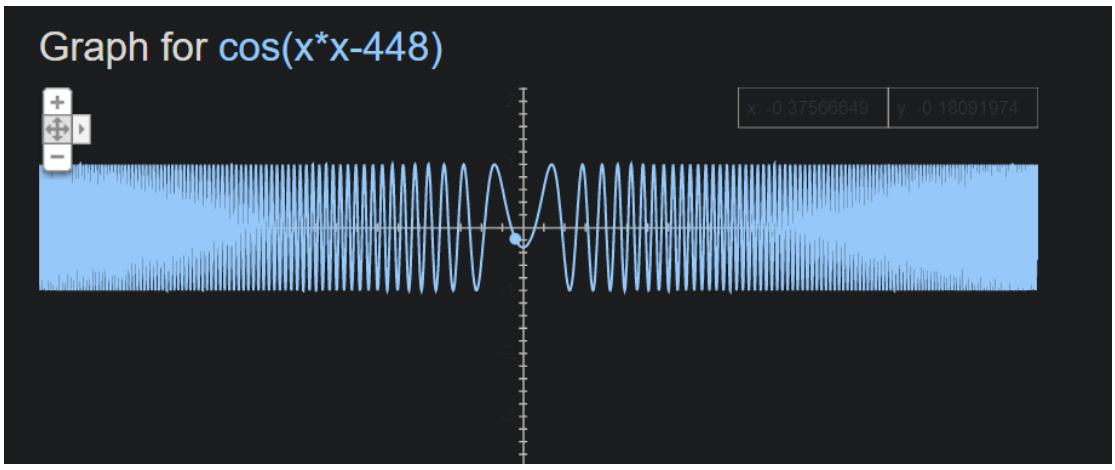
```
2 references | 2 references | 1 reference | 1 reference
const int limit_of_operator = 8 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 20 , maximum_variable_value = 300 , minumum_variable_value = -300 ;
10 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.001 ;
```



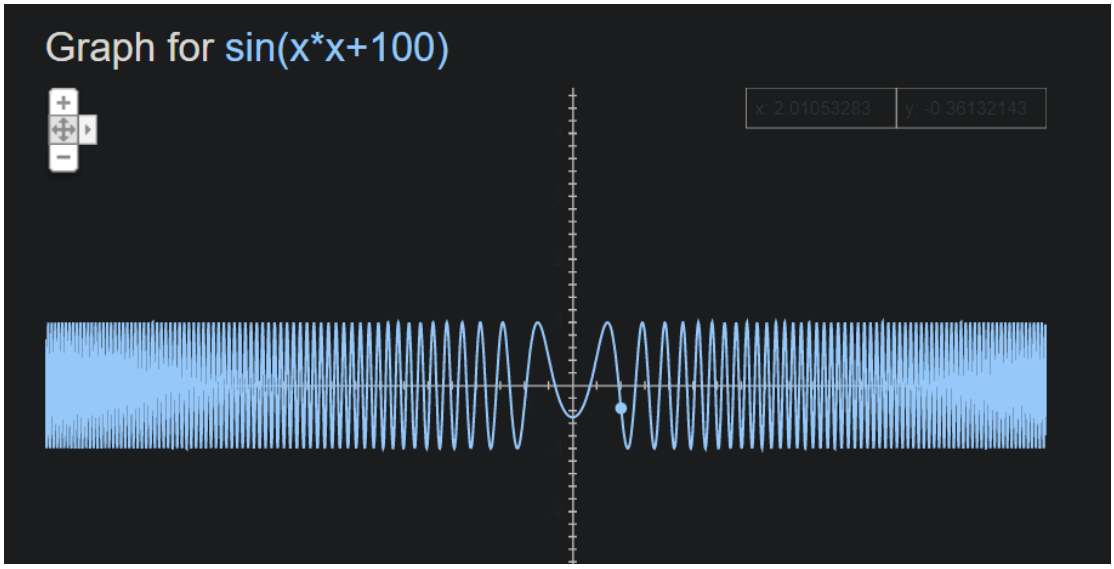
Result function:



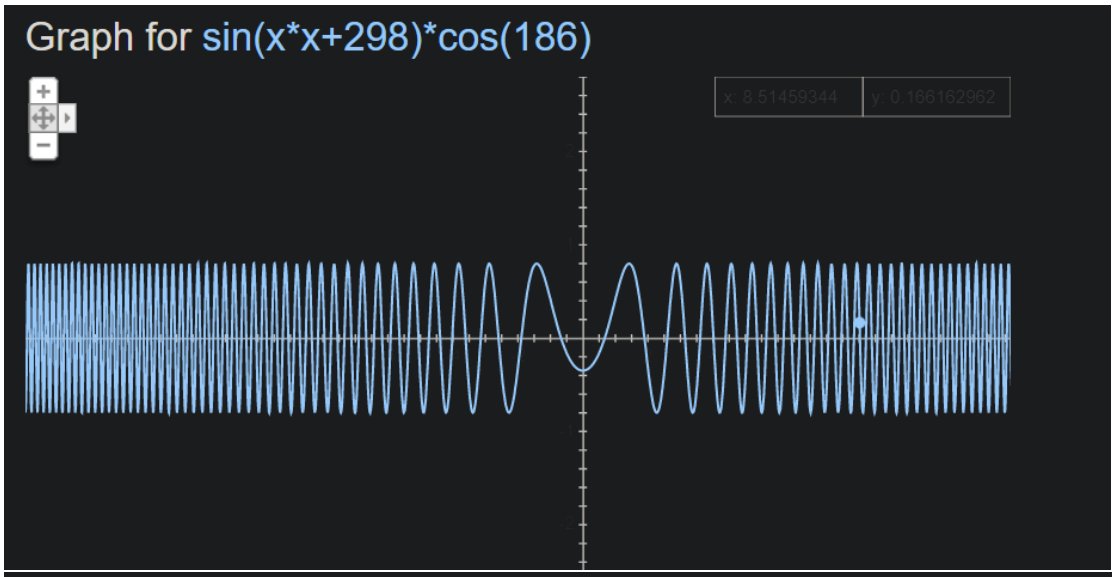


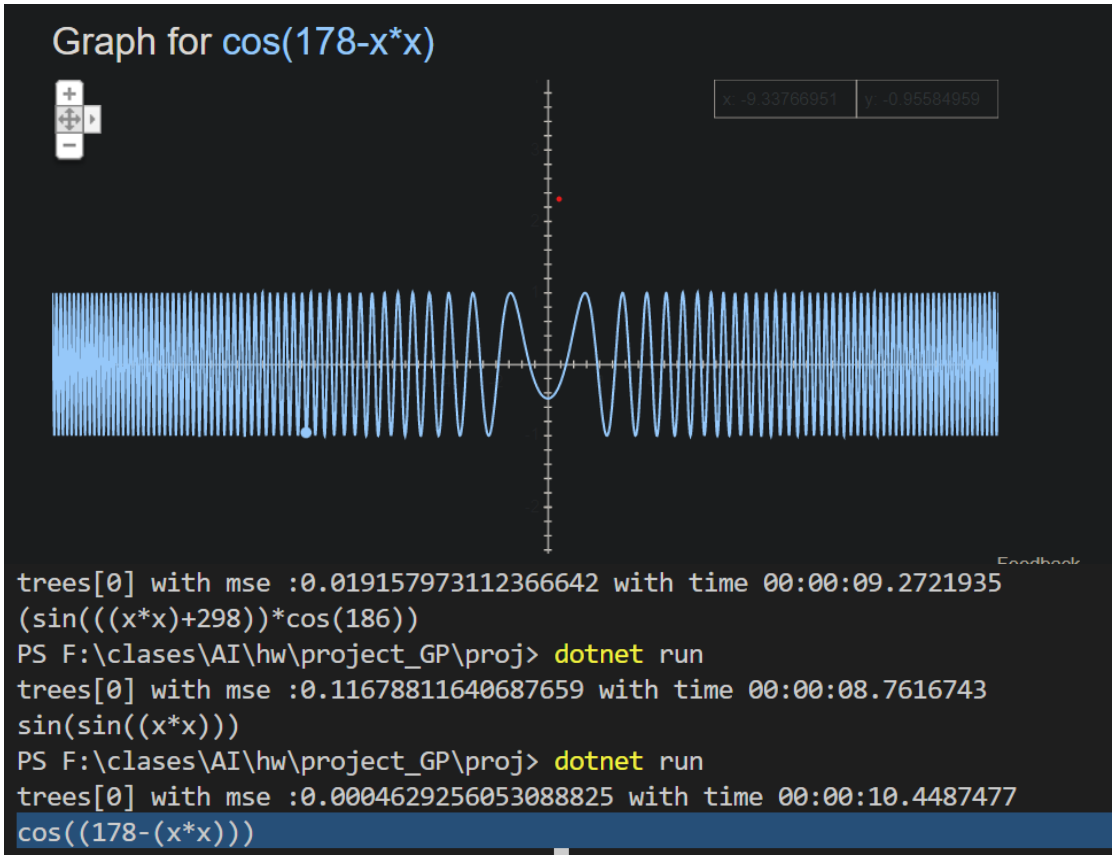


Target functin:



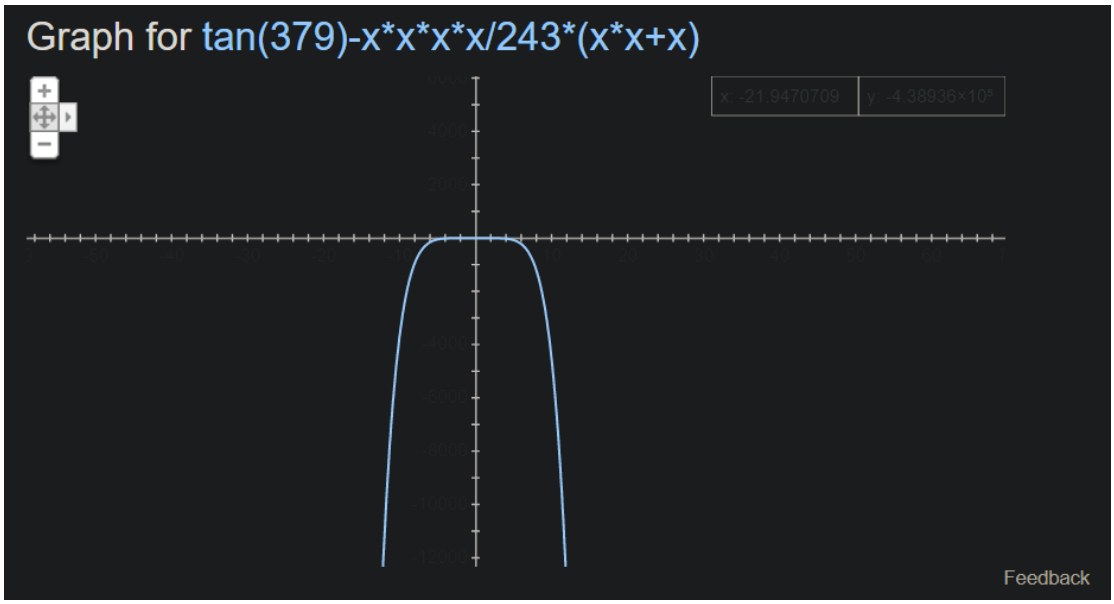
Result function:

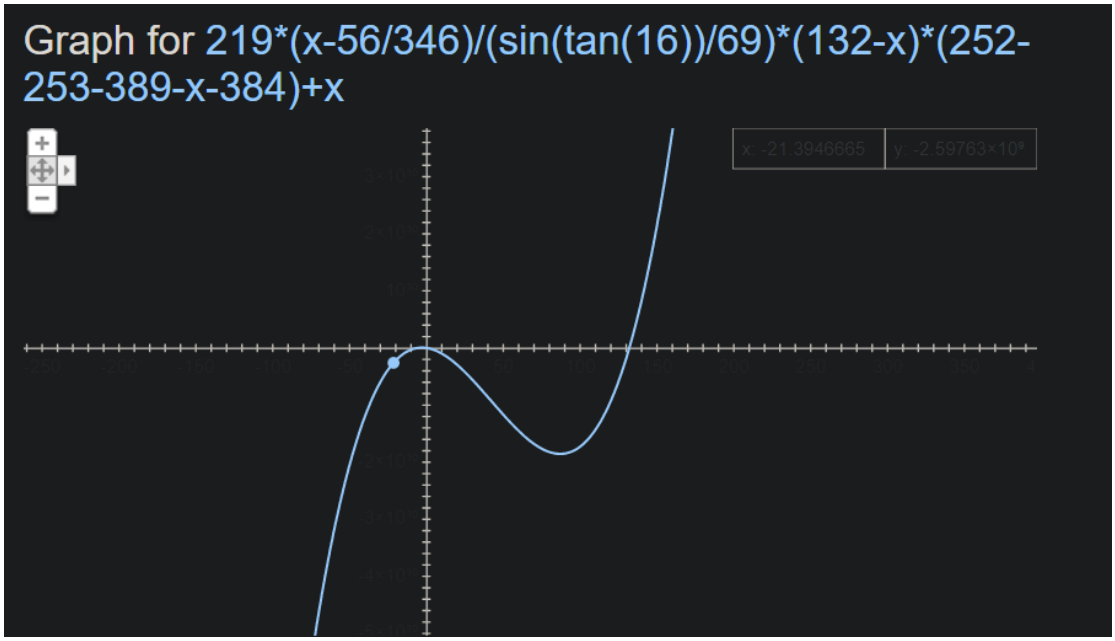




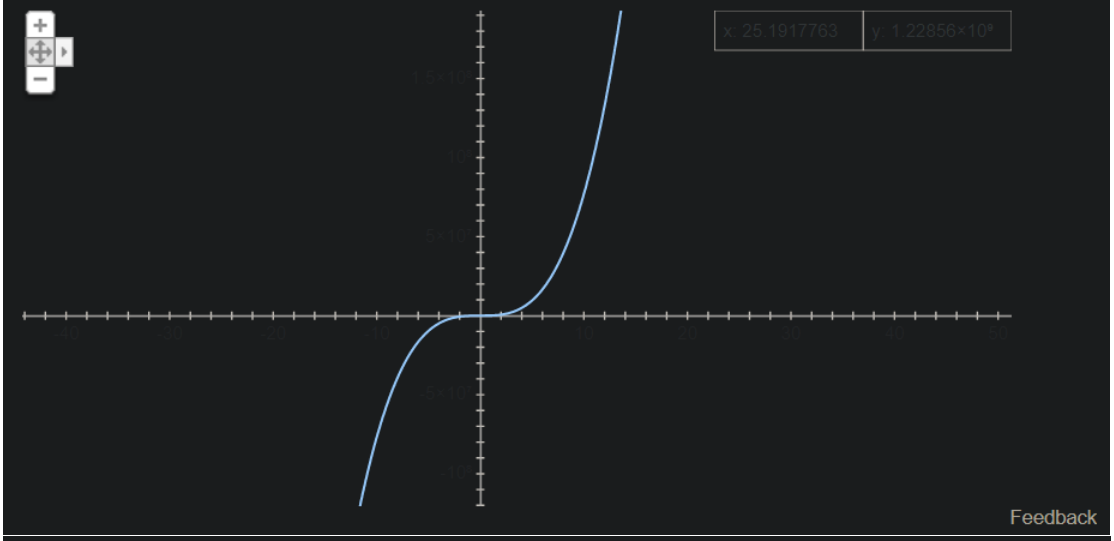
Target function:

Result function:

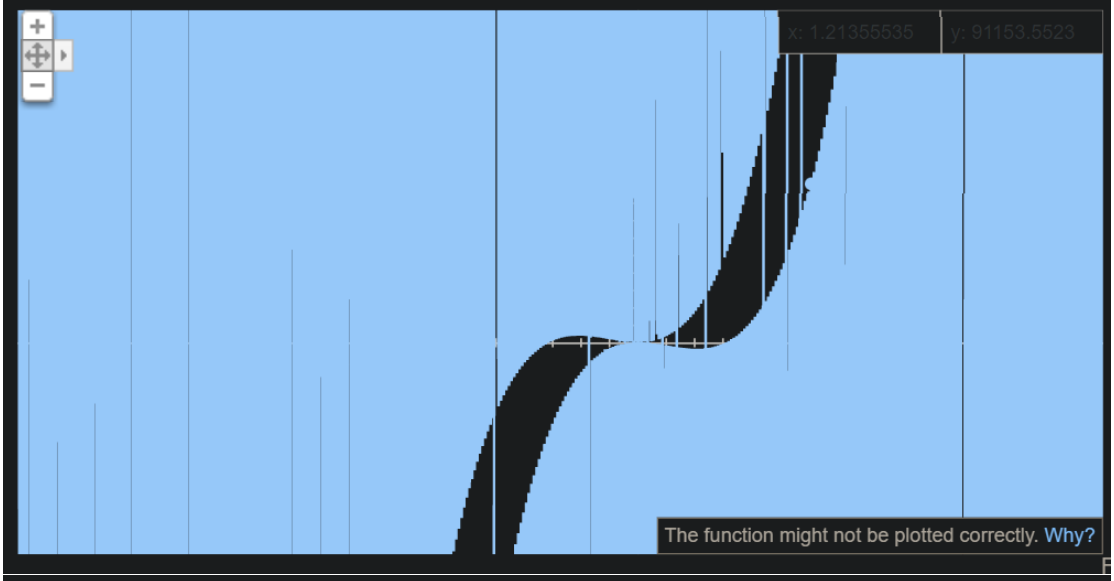




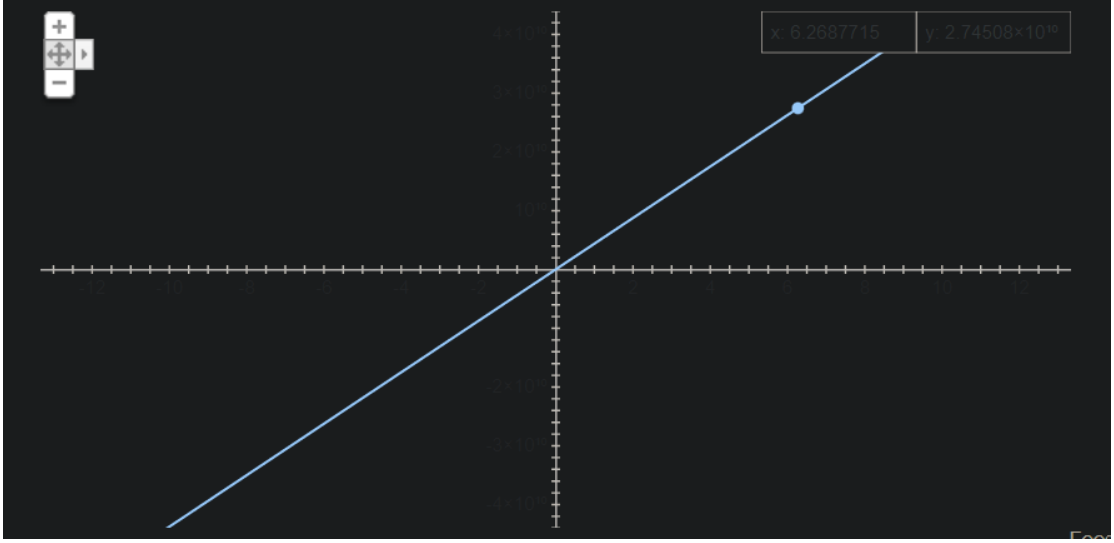
Graph for 499*x*x*154*x



Graph for (214/cos(339*x)+x*345)*x*x*305+359



Graph for (1/tan(355)*x-cos(98)*136)*321*411



```
trees[0] with mse :1.6128472772108827E+22 with time 00:00:11.7794435
(tan(379)-(x*(x*((x*(x/243))*((x*x)+x)))))
PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :8.090396093426554E+22 with time 00:00:09.3518249
(((((219*(x-(56/346)))/(sin(tan(16))/69))*(132-x))*(((252-253)-389)-(x-384)))+x)
```

```
trees[0] with mse :3.6329823238654544E+22 with time 00:00:13.6628420
((499*((x*x)*154))*x)
PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :1.4992429893089722E+23 with time 00:00:13.6232546
(((((214/cos((339*x)))+(x*345))*(x*x))*305)+359)
PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :4.6413912026901296E+23 with time 00:00:14.8477813
((249*((((409/(cot(sin(x))*386))+(445/tan(404)))*(190-(sin(252)+x)))*72))*212)
PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :1.883176352362314E+23 with time 00:00:10.1605145
((((cot(355)*x)-(cos(98)*136))*321)*411)
```

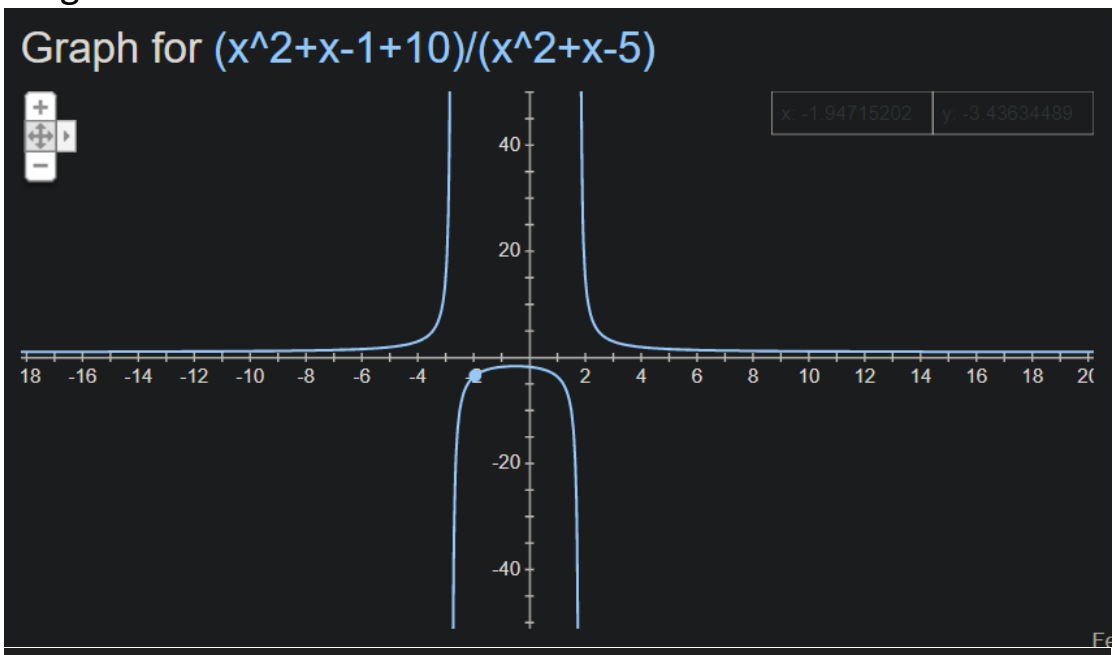
Target function:

```
2 references | 2 references | 1 reference | 1 reference
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 1 reference | 2 references
const int test_case = 20 , maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```

Result function:

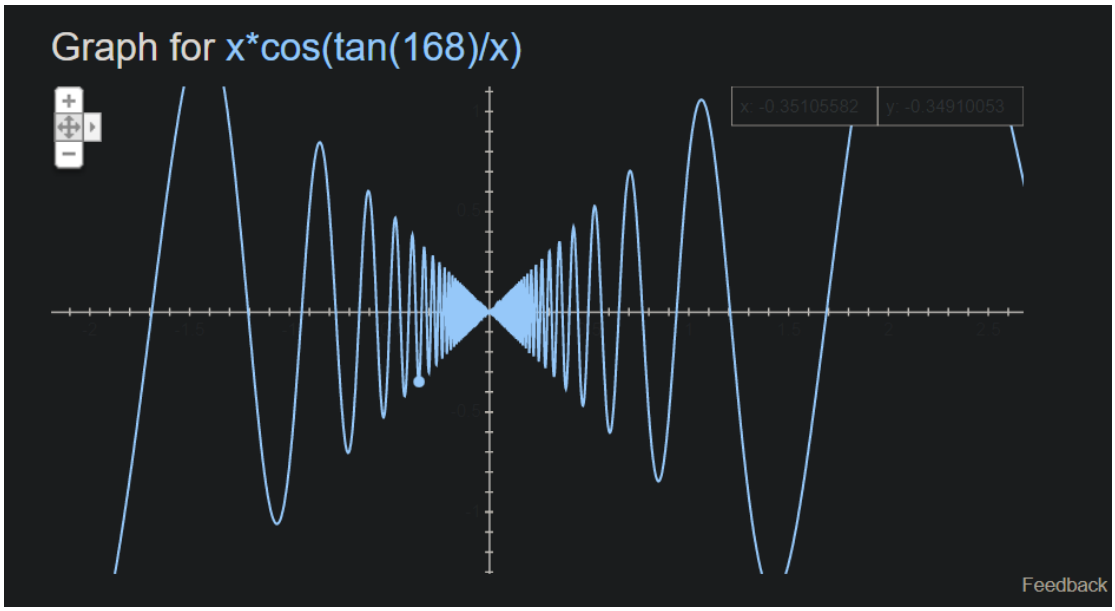
```
find best function with mse: 0 in 298 step with duration :00:00:02.7472753
(x+(x*(x*x)))

find best function with mse: 0 in 856 step with duration :00:00:07.6204864
(((x*x)*x)+x)

find best function with mse: 0 in 649 step with duration :00:00:05.2192623
(x+((x*x)*x))
```

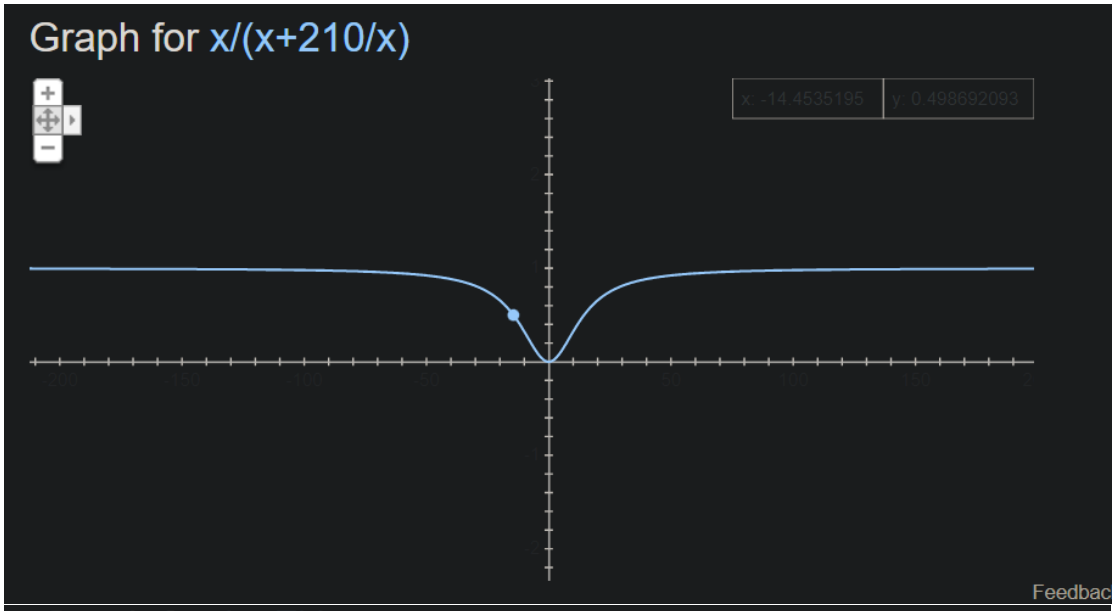
Target function:

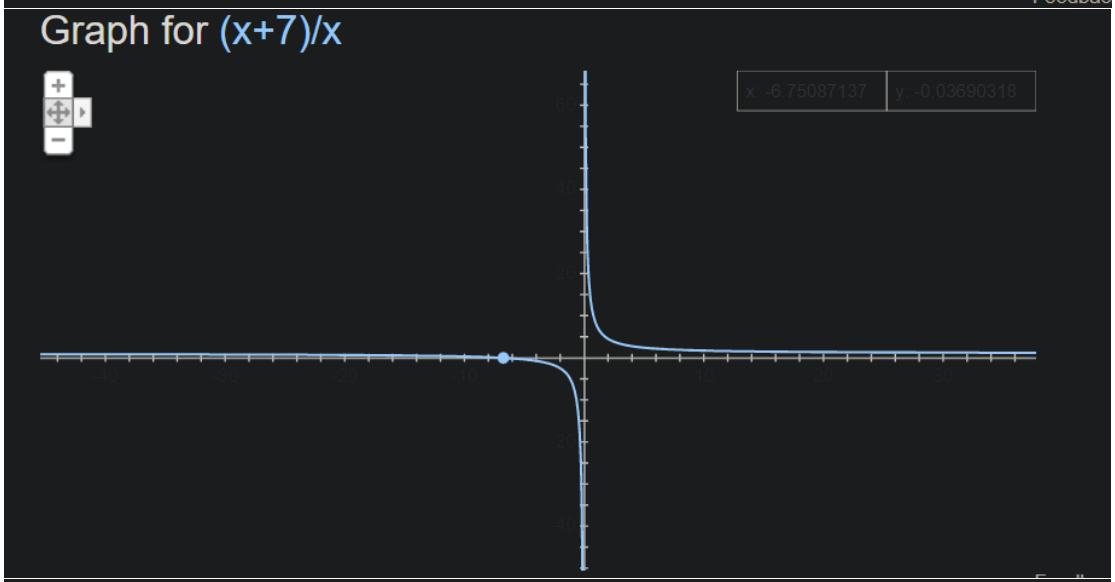



```
2 references | 2 references | 1 reference | 1 reference
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 20 , maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```

Result function:





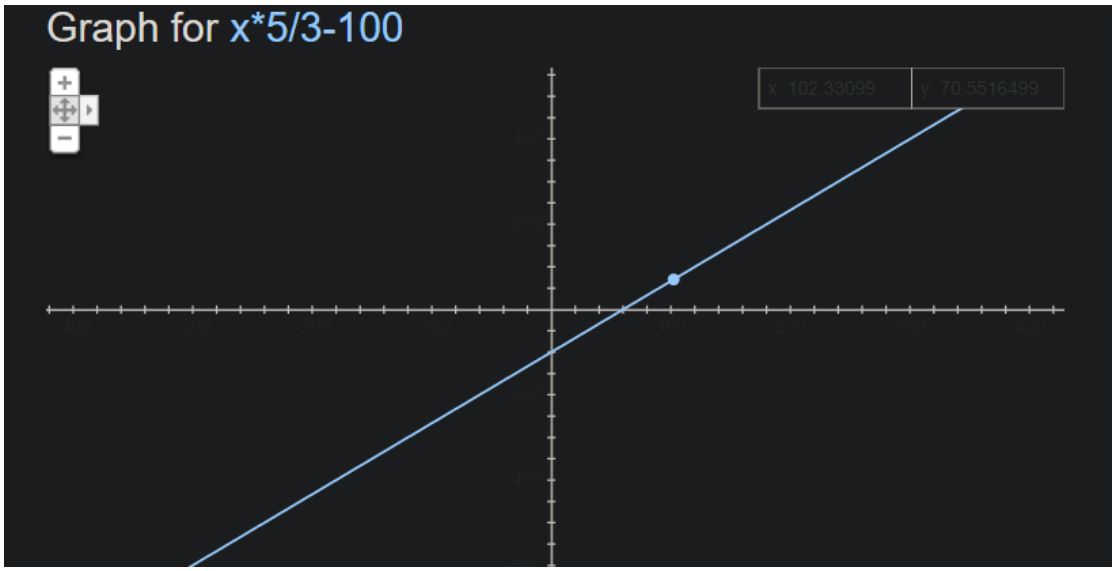



```
trees[0] with mse :0.005040088389333104 with time 00:00:10.2867476
(xcos((tan(168)/x))

trees[0] with mse :0.002300313355680405 with time 00:00:12.7877030
(x/(x+(210/x)))

trees[0] with mse :5.9056507324918625E-05 with time 00:00:14.8804461
((x+7)/x)
```

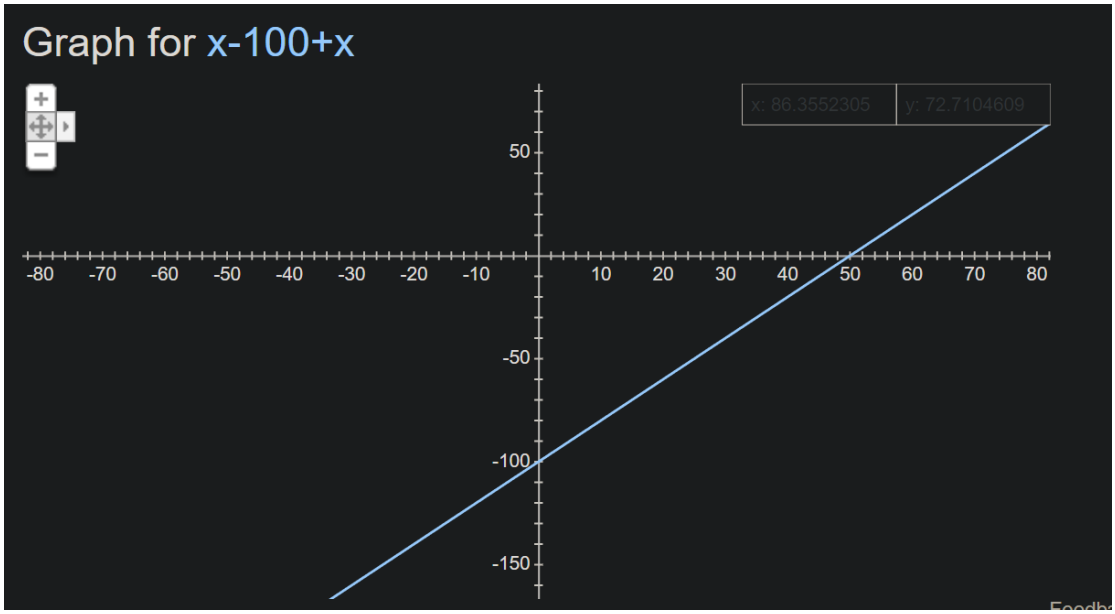
همانطور که میبینیم ، بعضی اوقات دقیق عمل نمیکنه پس تعداد تست کیس را زیاد میکنیم(تست کیس ها به صورت رندوم انتخاب میشوند) و نتیجه را گزارش میکنیم

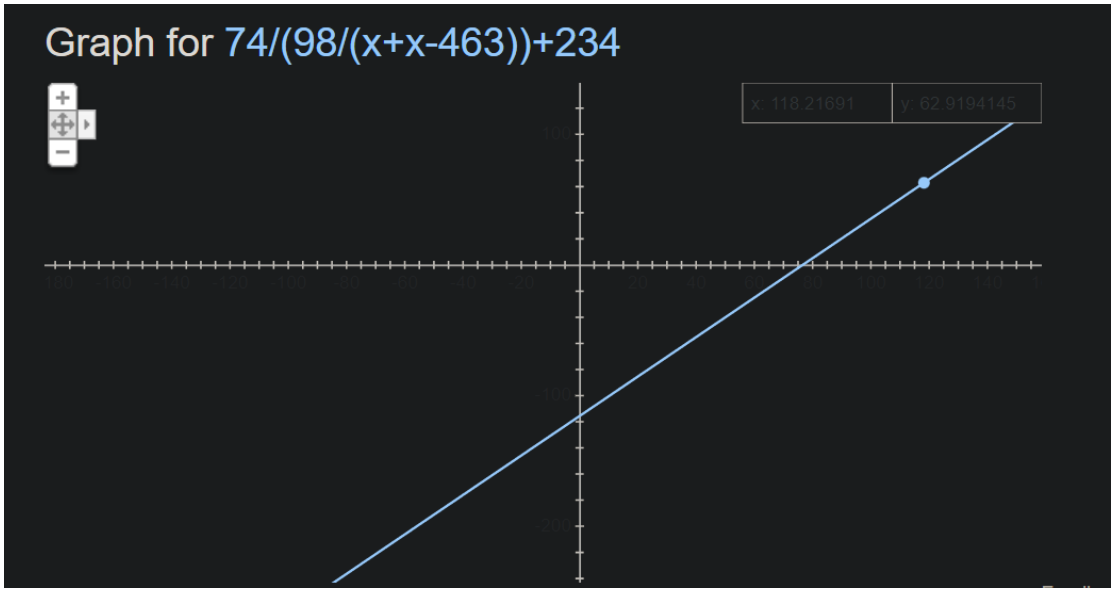
Target function:

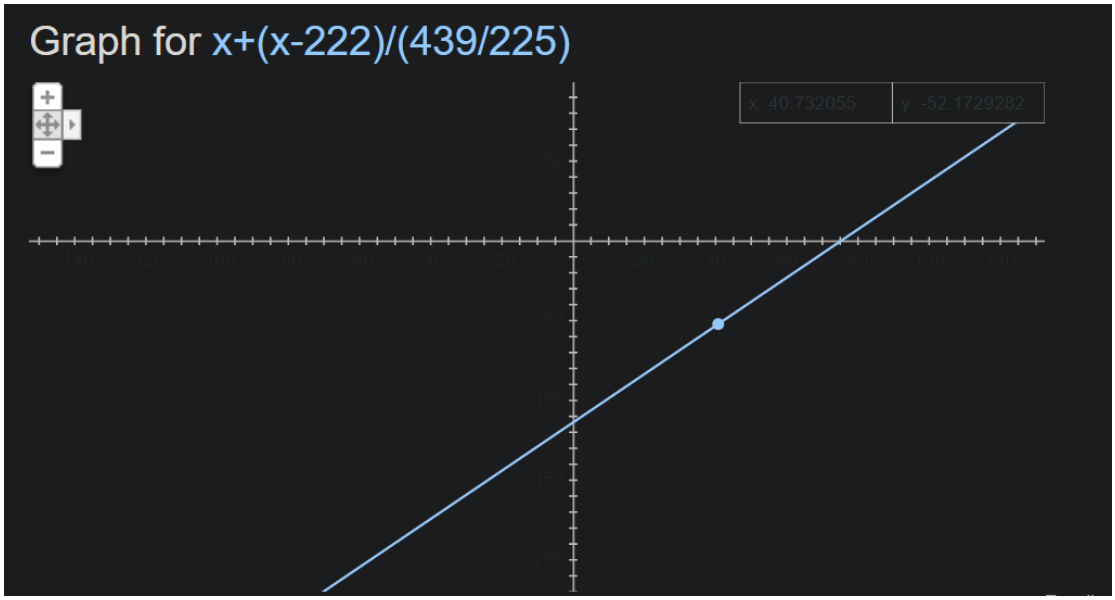
```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```



Result function:



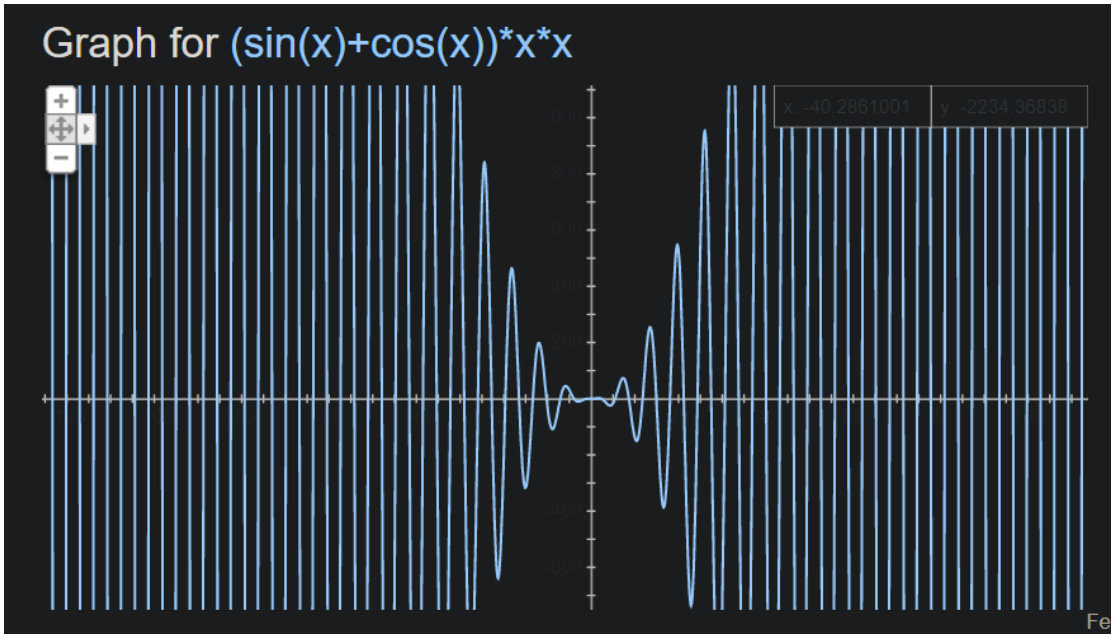




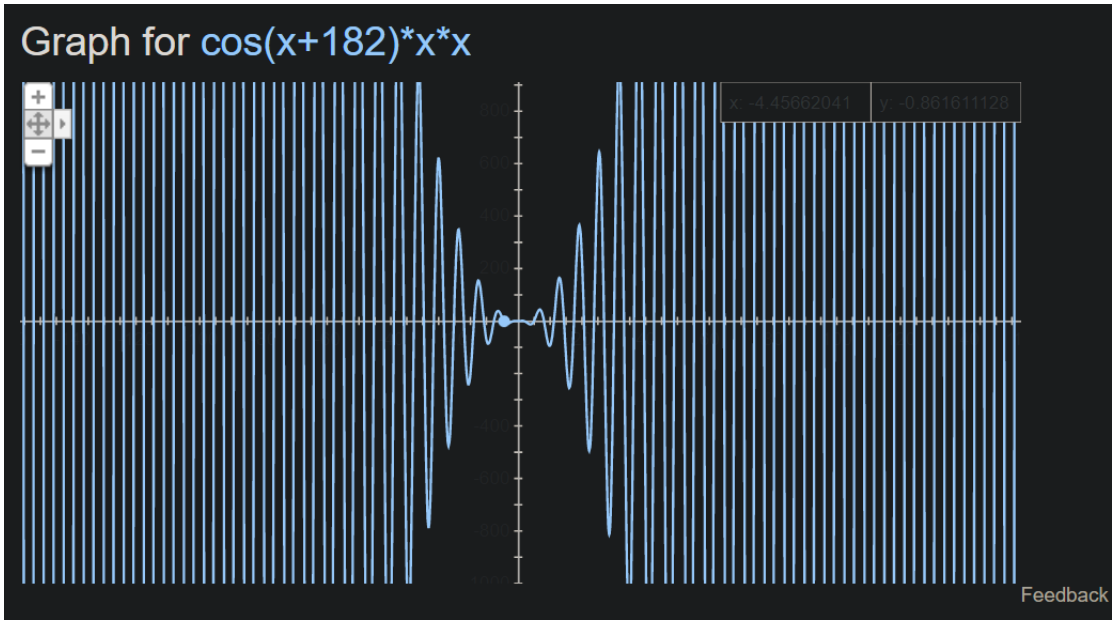

```
trees[0] with mse :3301.125 with time 00:00:45.1630917
((x-100)+x)

trees[0] with mse :825.6986151603498 with time 00:00:47.4350653
((74/(98/(x+(x-463))))+234)

trees[0] with mse :915.3069069406031 with time 00:00:50.9228419
(x+((x-222)/(439/225)))
```

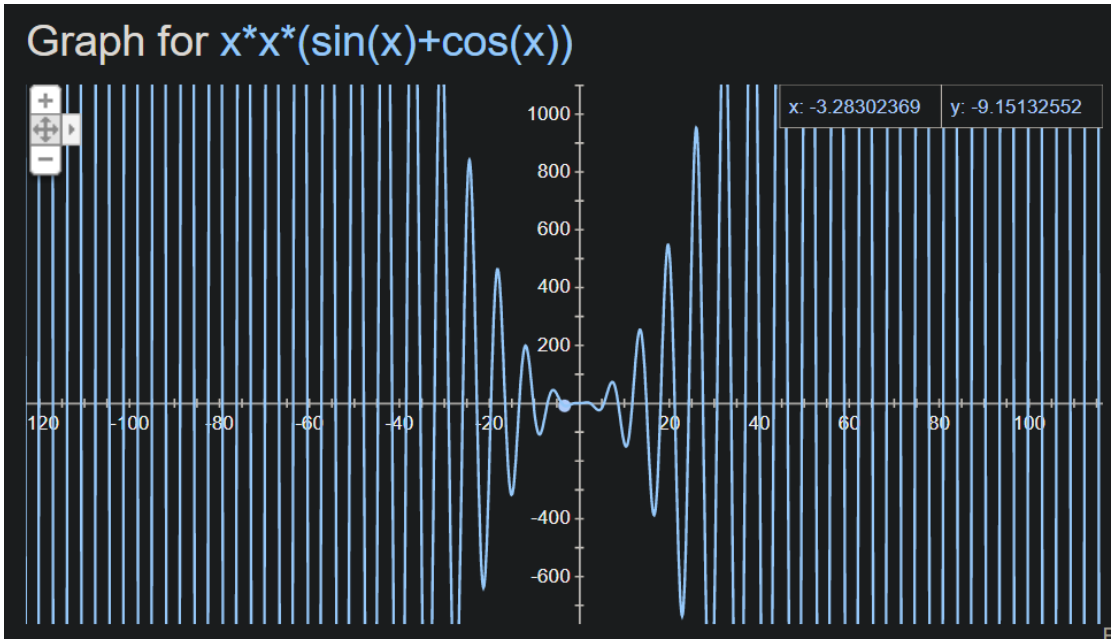
Target function:

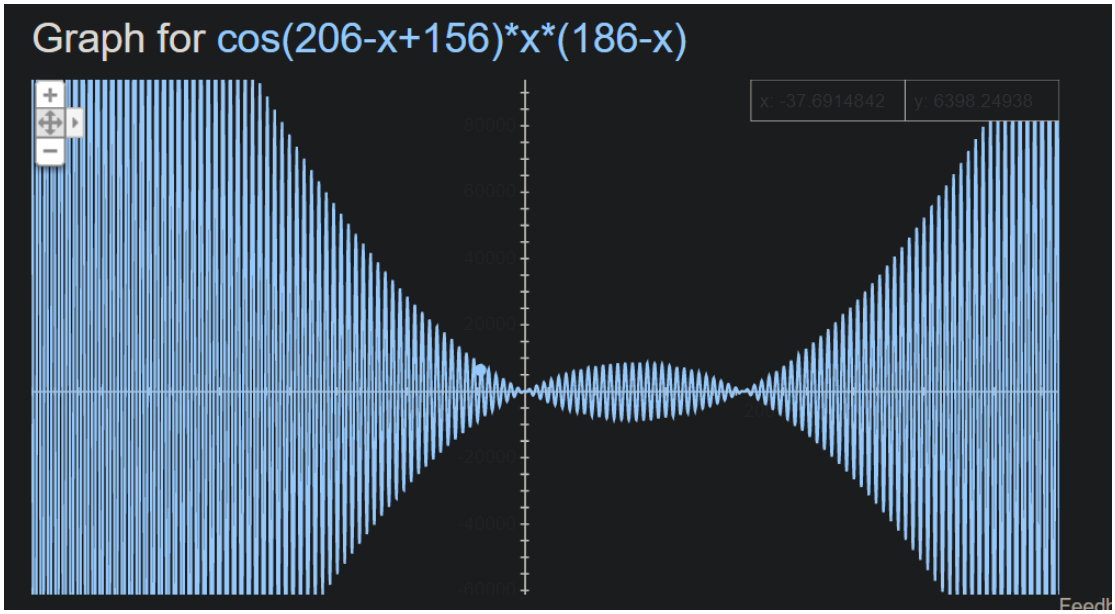
```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```



Result function:



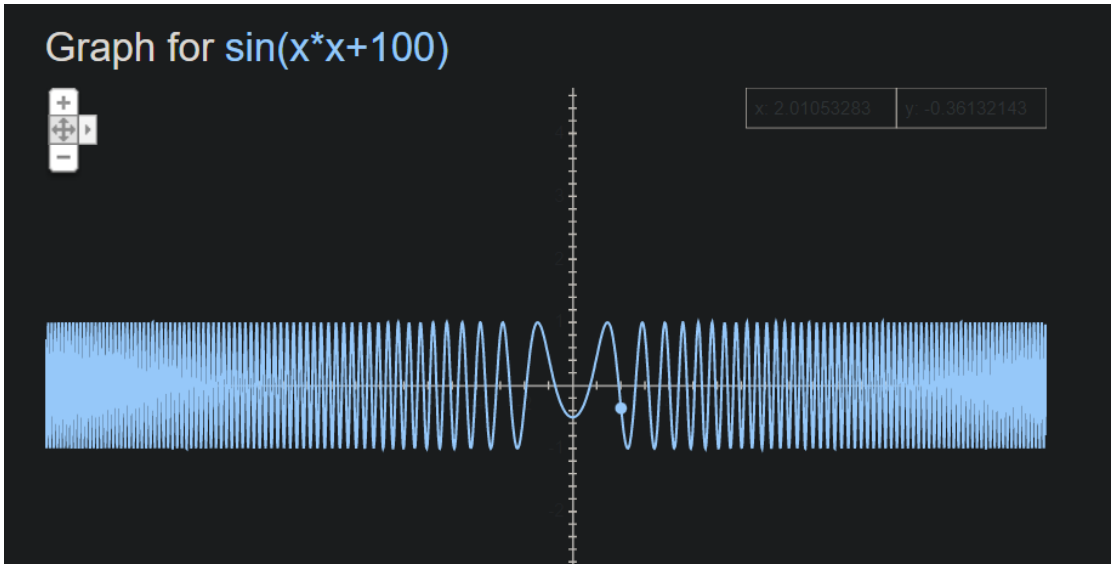




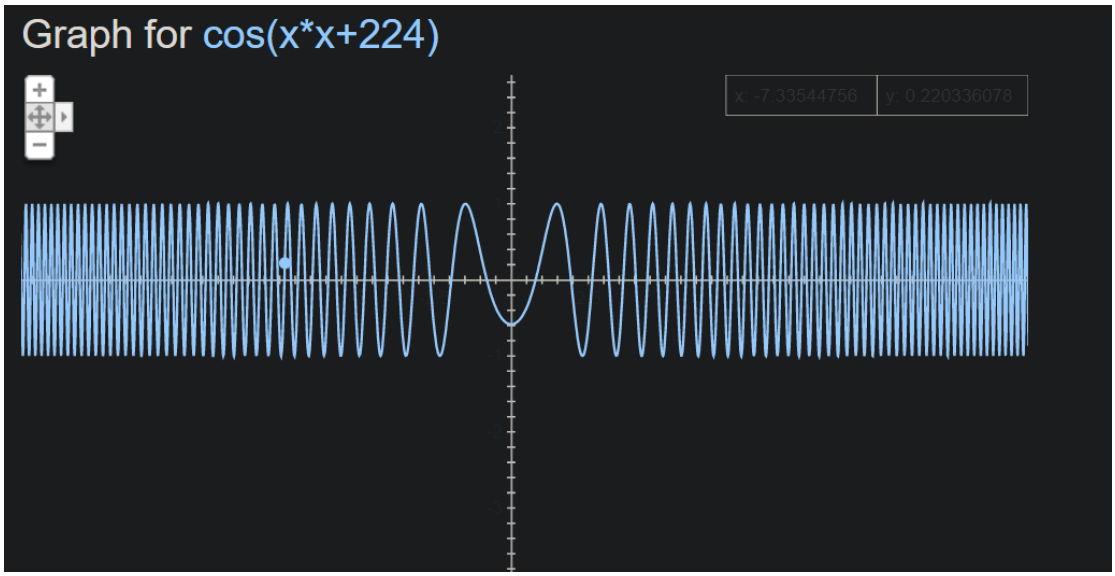

```
trees[0] with mse :593661476.907695 with time 00:00:51.8706639
(cos((x+182))*(x*x))

find best function with mse: 5.3095061042752745E-24 in 808 step with duration :00:00:52.7607816
((x*x)*(sin(x)+cos(x)))

trees[0] with mse :630792399.0565963 with time 00:00:55.5661086
(cos(((206-x)+156))*(x*(186-x)))
```

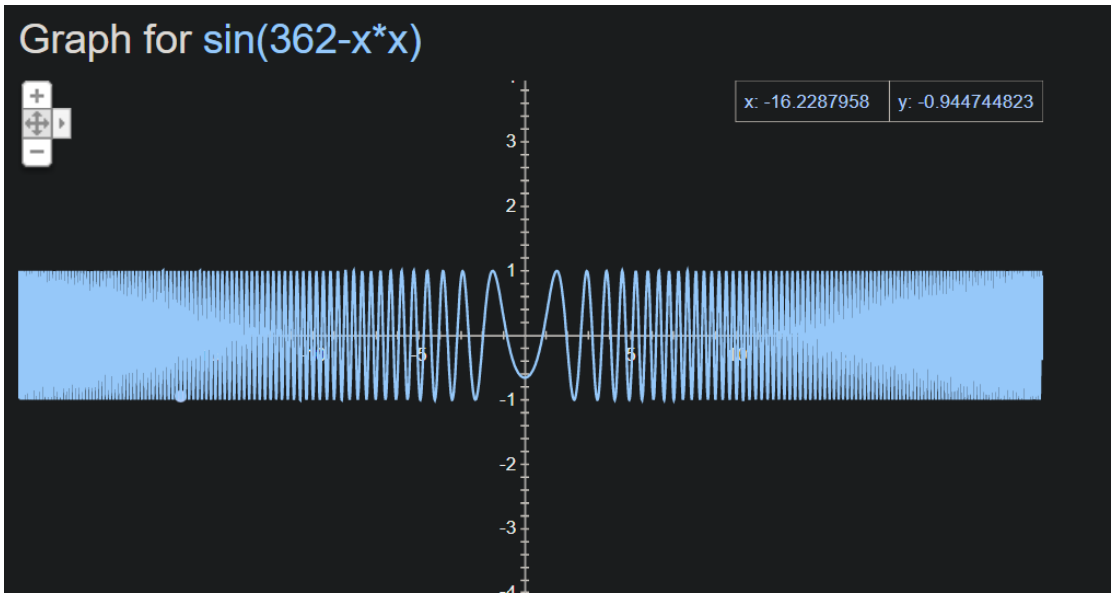
Target functin:

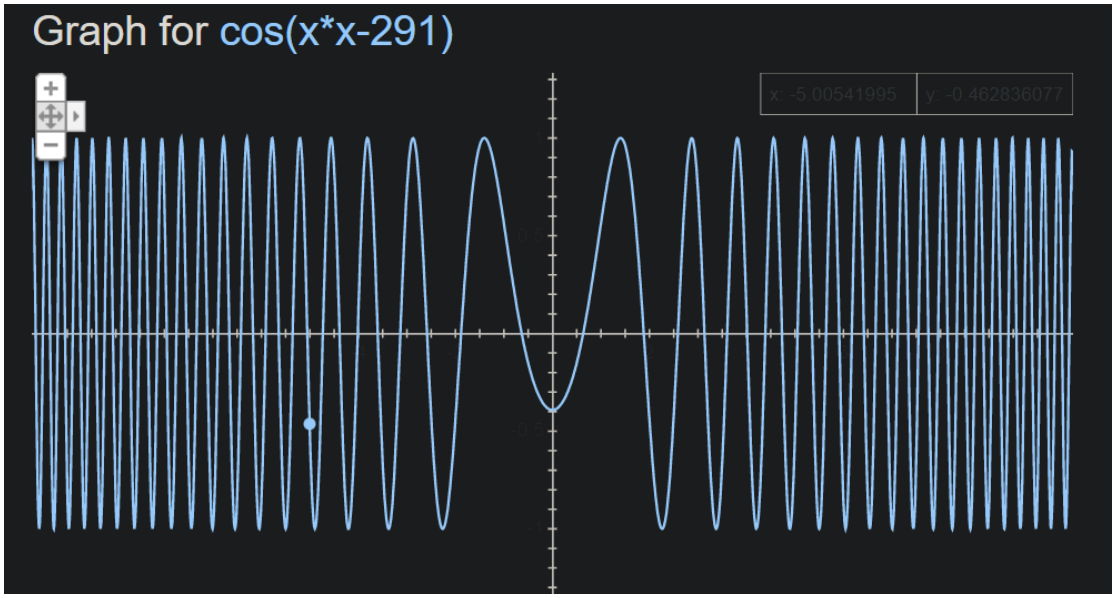
```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```



Result function:



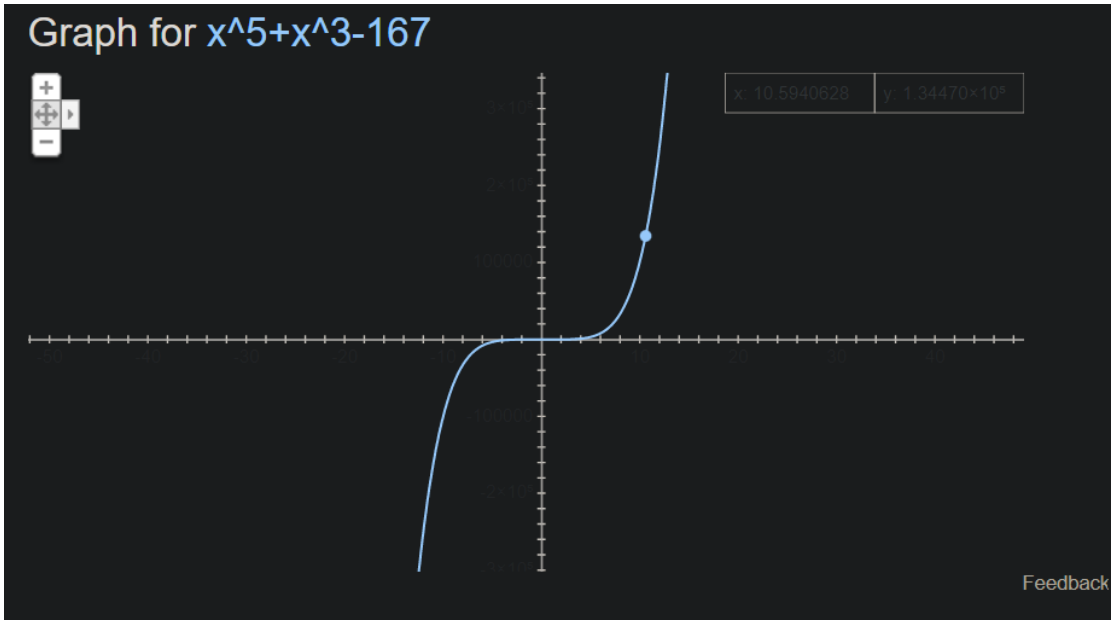




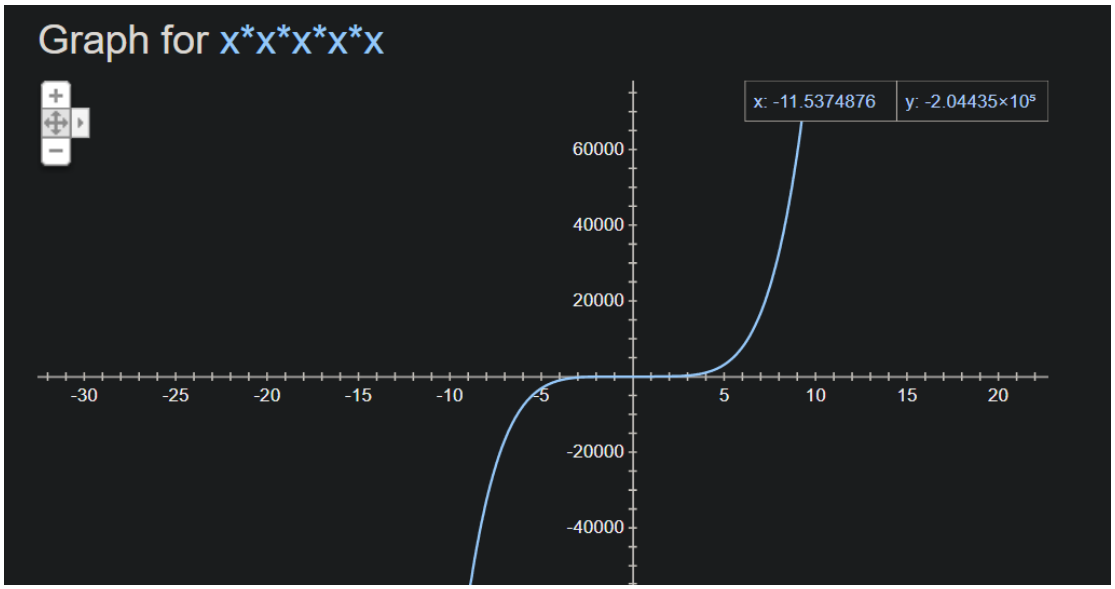

```
trees[0] with mse :0.00464596833893898 with time 00:00:53.2039608
cos(((x*x)+224))

trees[0] with mse :0.017554716330073956 with time 00:00:47.3401682
sin((362-(x*x)))

trees[0] with mse :0.008933207752794608 with time 00:00:55.1771651
cos(((x*x)-291))
```

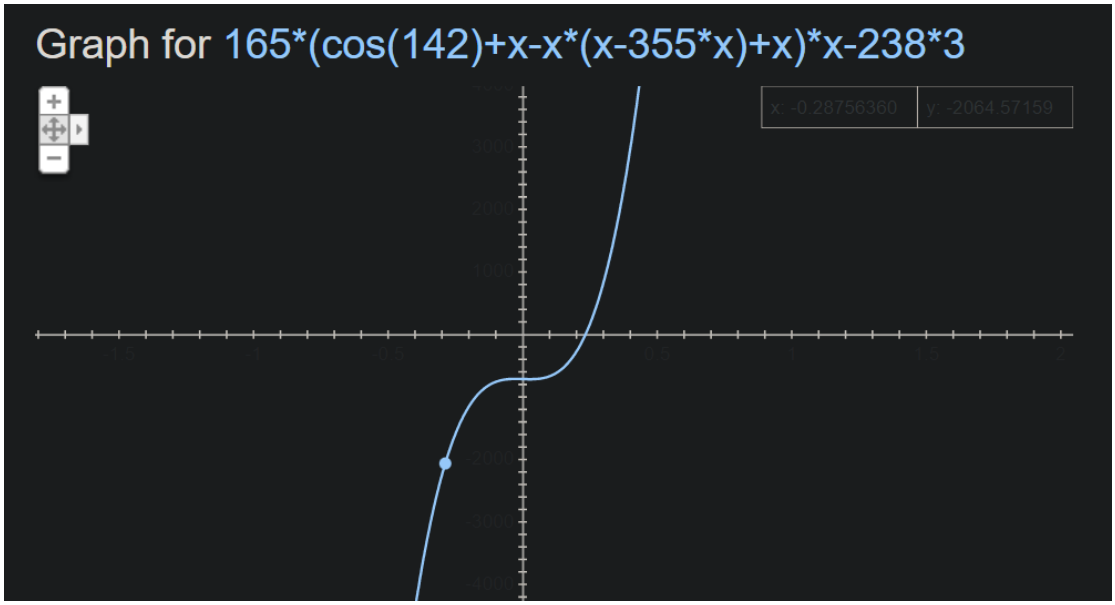
Target function:

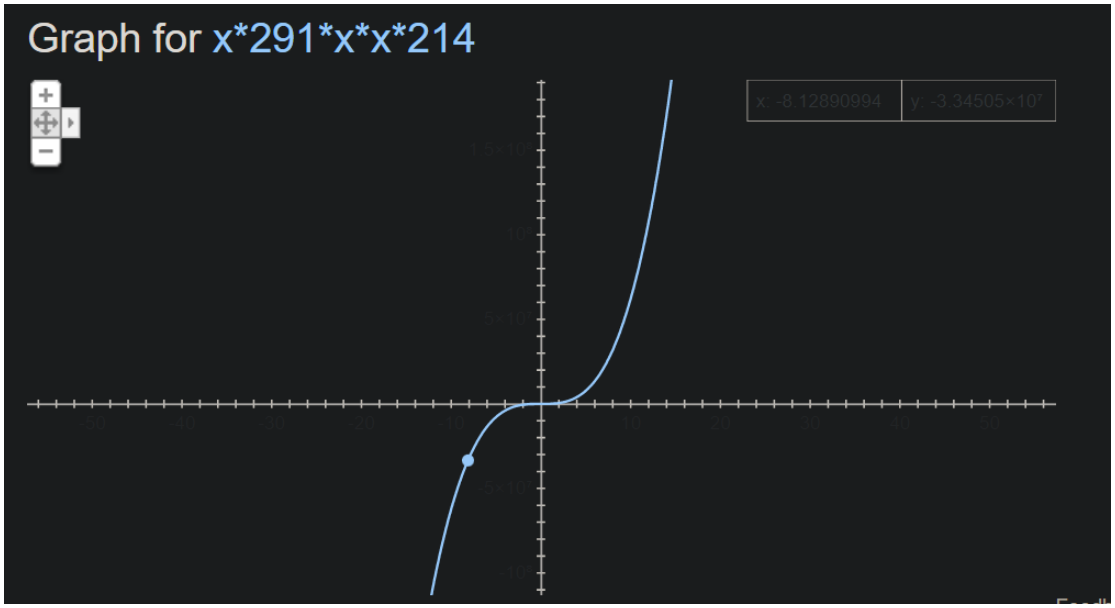
```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```



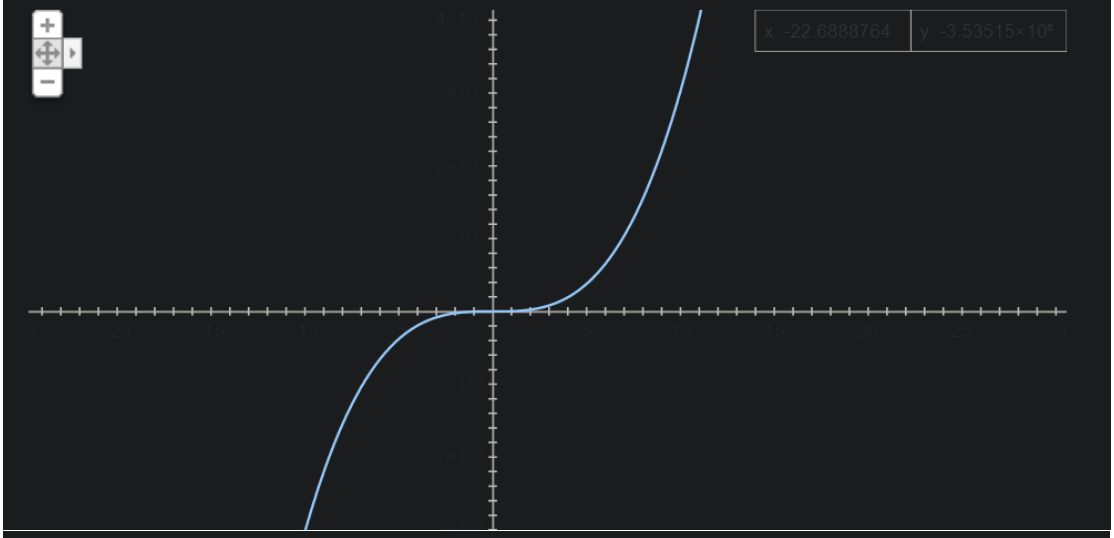
Result function:



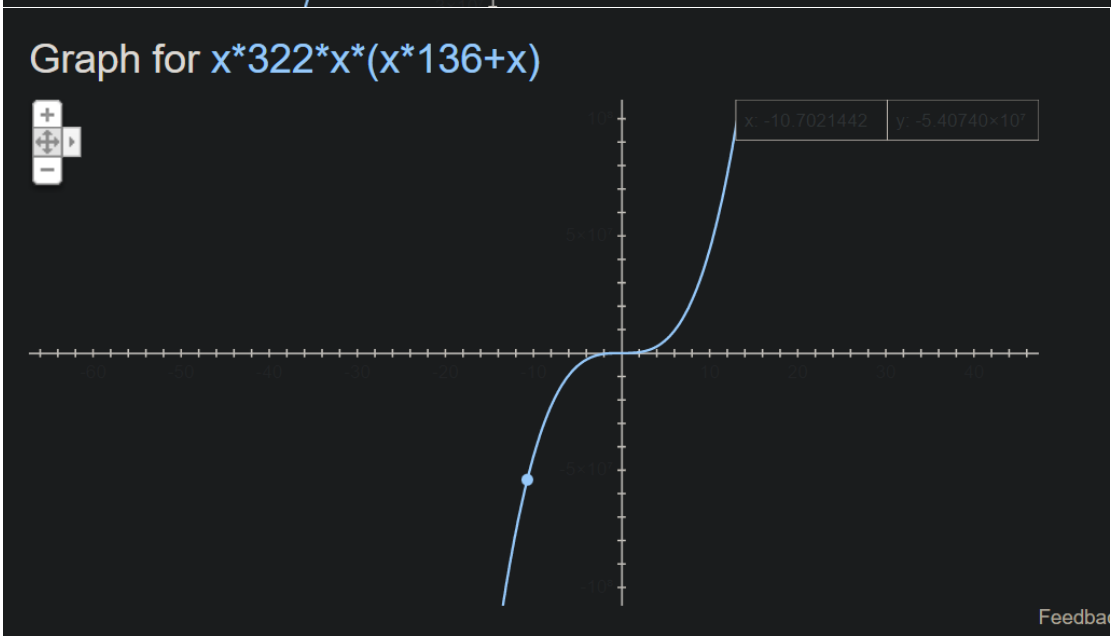

Graph for 165*(cos(142)+x-x*(x-355*x)+x)*x-238*3
x: -0.28756360
y: -2064.57159


Graph for x*291*x*x*214
x: -8.12890994
y: -3.34505×10⁷

Graph for x*177*171*x*x



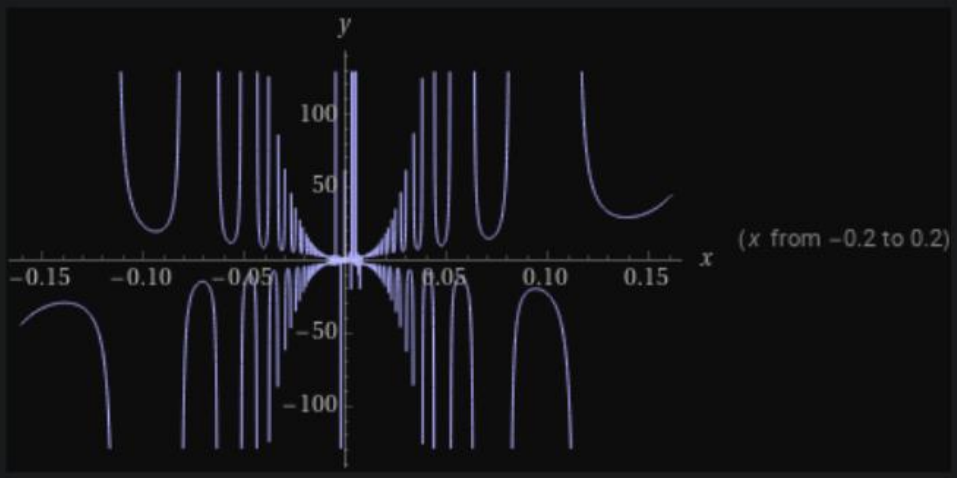
Graph for x*322*x*(x*136+x)

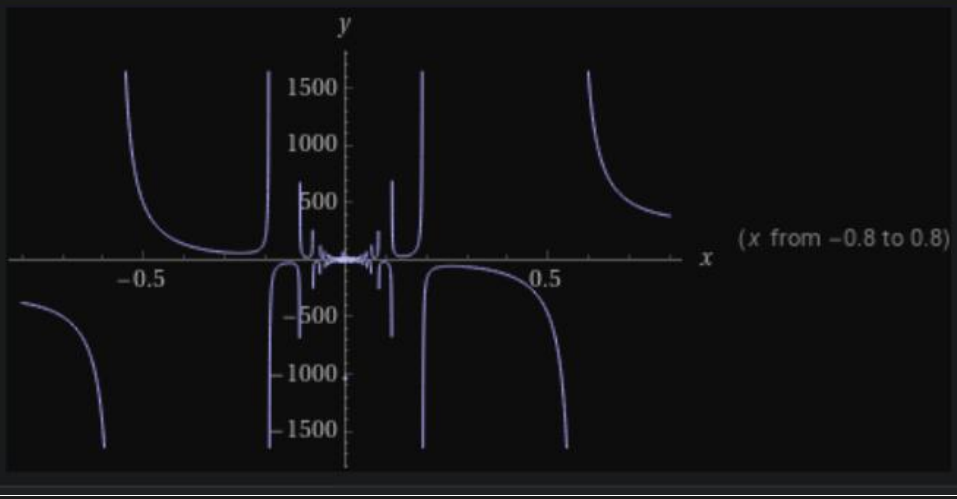


Feedba

plot $\frac{x}{\frac{1}{208}\cos\left(\frac{246}{(275-((x-x)+x))\,x}\right)}$

Plots






```
trees[0] with mse :86184173734181.6 with time 00:00:58.2976938
((x*(x*x))*(x*x))

trees[0] with mse :5.05312704773569E+22 with time 00:01:12.3649928
((165*(((cos(142)+x)-((x*(x-(355*x)))+x))*x))-(238*3))

trees[0] with mse :1.7037928707088502E+22 with time 00:01:19.6935807
(((x*291)*(x*x))*214)

PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :1.5734381400858094E+23 with time 00:01:12.1107652
((((x*177)*171)*x)*x)

trees[0] with mse :7.113527585824309E+22 with time 00:01:05.4035750
((x*322)*(x*((x*136)+x)))

trees[0] with mse :3.1293956352298254E+23 with time 00:01:36.2570881
(x/(cos((246/((275-((x-x)+x))*x)))/208))
```

Target function:

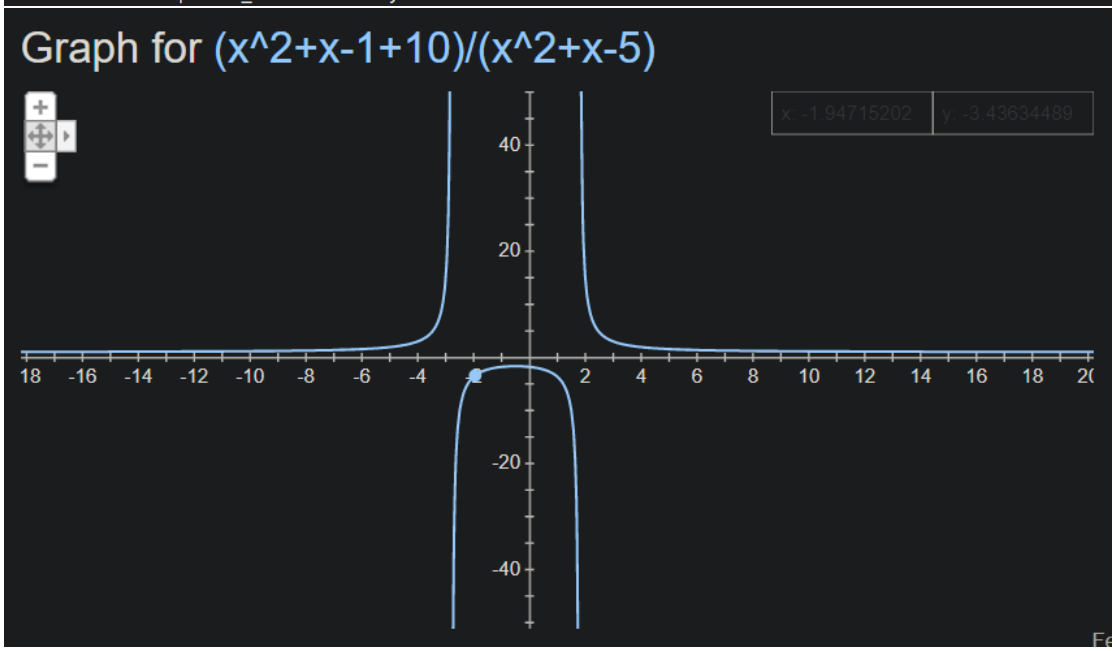
```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```

Result function:

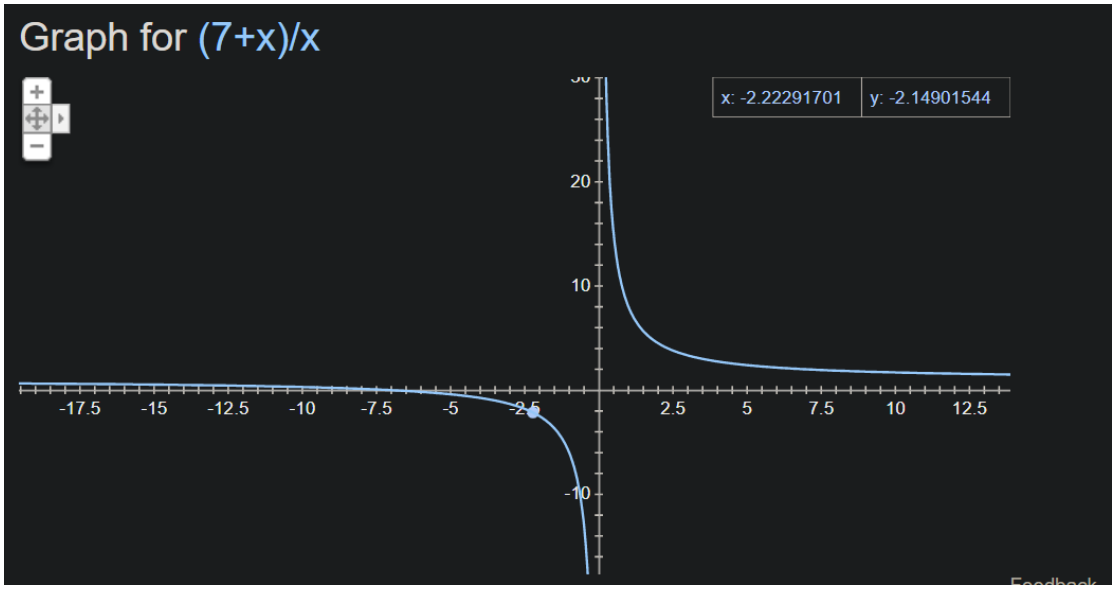
```
find best function with mse: 0 in 276 step with duration :00:00:07.8819896
((x*(x*x))+x)

trees[0] with mse :28289.1625 with time 00:00:40.7638240
(x*(x*x))

trees[0] with mse :24638.825 with time 00:00:32.3800036
(x*(x*x))

PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :1564.4258040343634 with time 00:00:40.8273433
(((((x-(x/(cos(206)+x)))*x)+x)*x)+x)

find best function with mse: 0 in 341 step with duration :00:00:12.3701322
((x*(x*x))+x)

find best function with mse: 0 in 69 step with duration :00:00:03.9051919
(x+((x*x)*x))
```

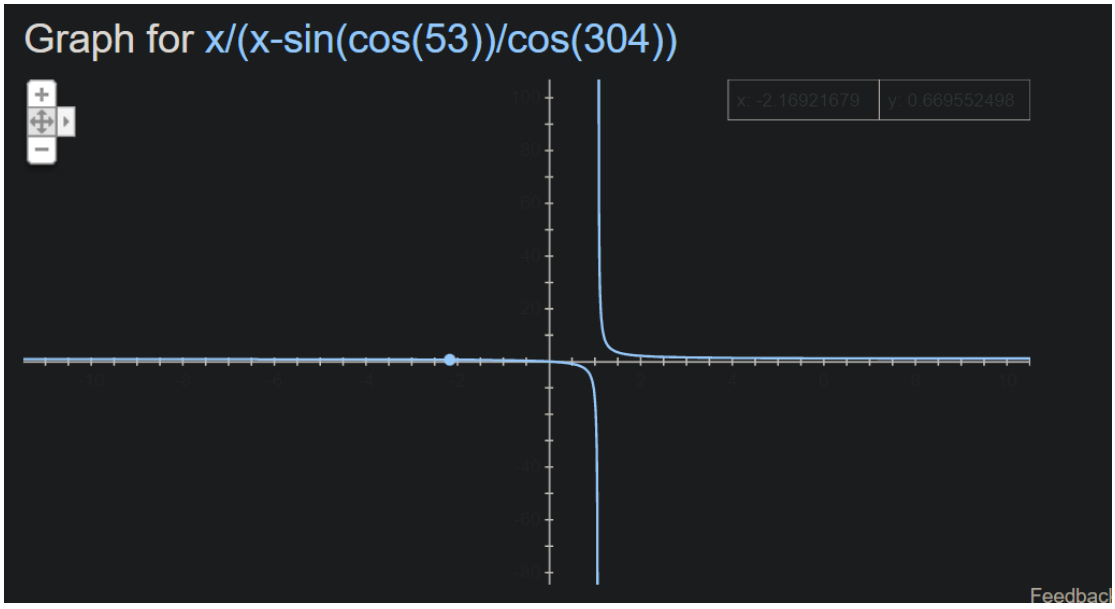
Target function:

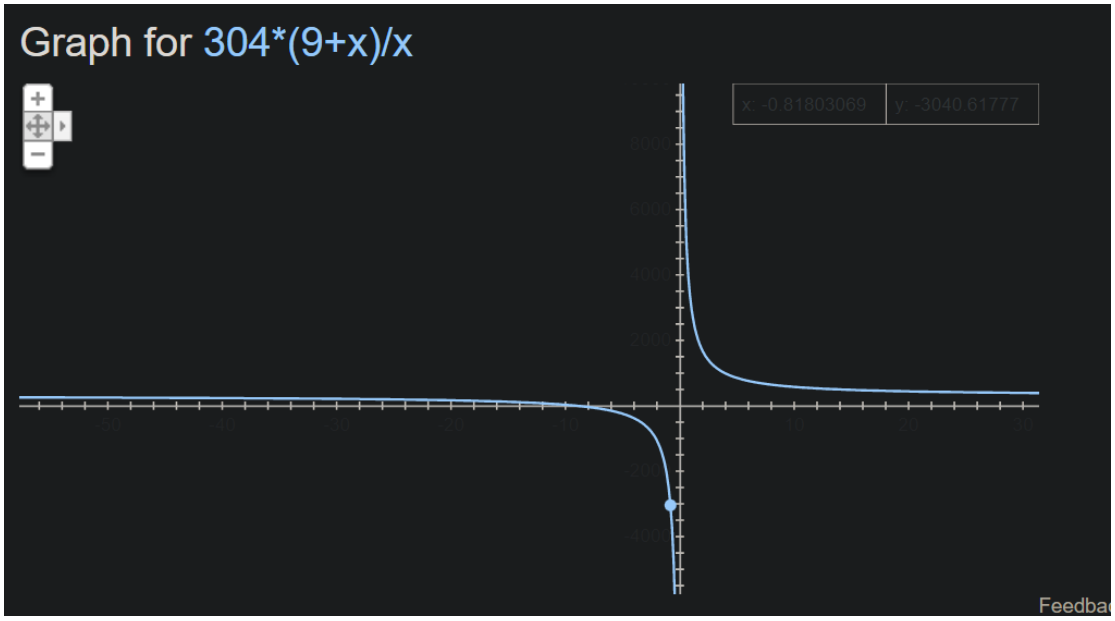
```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
9 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 15, worst_mse_percent = 10 ;//,best_mse_percent = 60 ;
1 reference
const double expected_mse =0.00001 ;
```

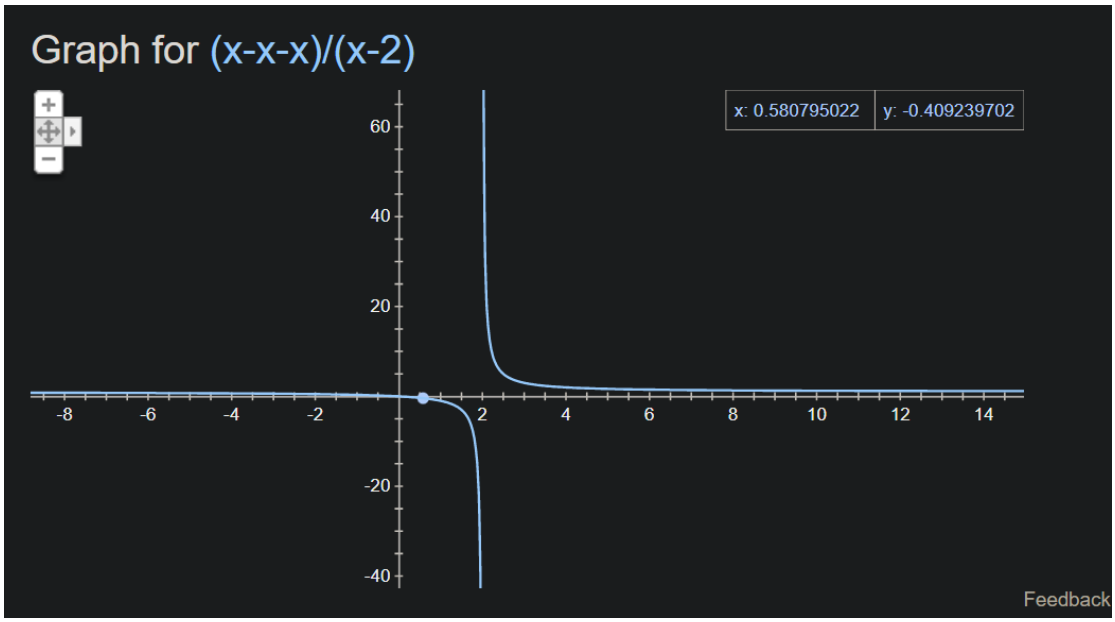


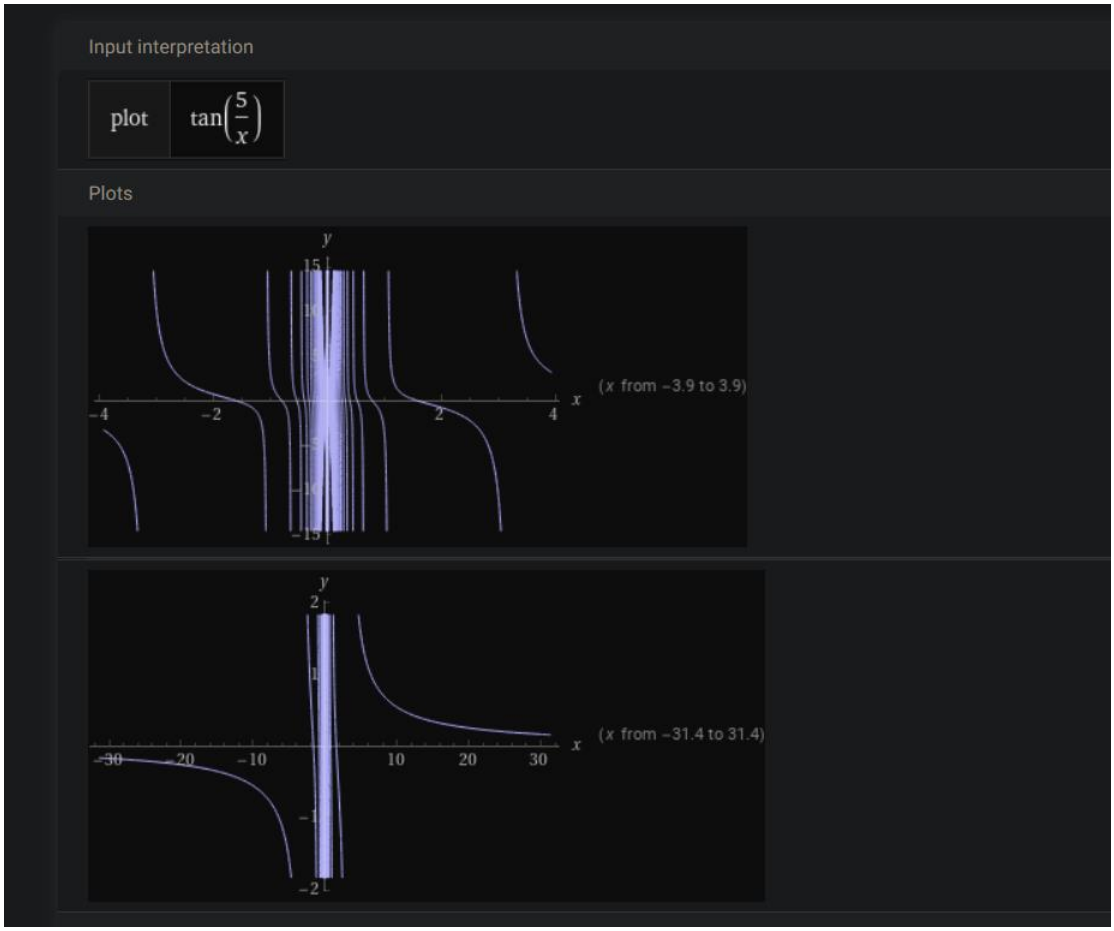
Result function:






Graph for 304*(9+x)/x
x: -0.81803069
y: -3040.61777
8000
6000
4000
2000
-50
-40
-30
-20
-10
10
20
30
-2000
-4000
Feedba


Graph for (x-x-x)/(x-2)
x: 0.580795022
y: -0.409239702
60
40
20
-20
-40
-8
-6
-4
-2
2
4
6
8
10
12
14
Feedback




```
trees[0] with mse :0.0133632439407715 69 with time 00:00:56.8559808
((7+x)/x)

trees[0] with mse :2.8588017131263874 with time 00:01:06.5723822
(x/(x-(sin(cos(53))/cos(304))))

trees[0] with mse :0.0011534822654696761 with time 00:00:42.5847430
((9+x)/x)

PS F:\clases\AI\hw\project_GP\proj> dotnet run
trees[0] with mse :0.0483210362543829 85 with time 00:00:50.8542017
((x-(x-x))/(x-2))

trees[0] with mse :0.0029215391605199956 with time 00:01:02.0160390
((x+10)/x)

trees[0] with mse :1.0679227487385967 with time 00:01:00.6770317
tan((5/x))
```

بقیه توابع نتیجه بهتری داشتن ولی تابع اخر خیلی خوب نبود

-تابع جهش را تغییر دادم(بعد از تولید تابع کراس اور شده با احتمال کم به تغیر یک گره اتفاقی)

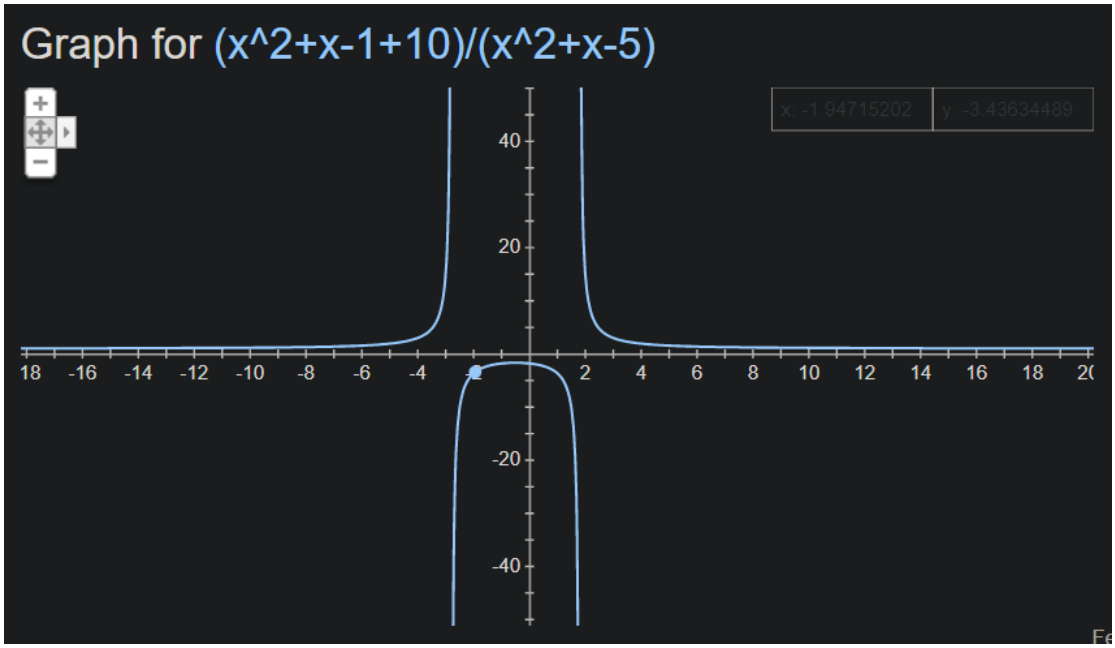
- تابع تولید مثل را تغیر دادم(چون تعداد کراس اور بین نسل خوب به

نسبت تعدادشان زیاد بود و بهبود هایی برای خود تابع تولید
مثل انجام دادم برای مثال برای تابع های تکراری پشت سر هم که کم
میشد در اخر درخت های تصادفی ایجاد کردم تا تعداد ثابت بماند

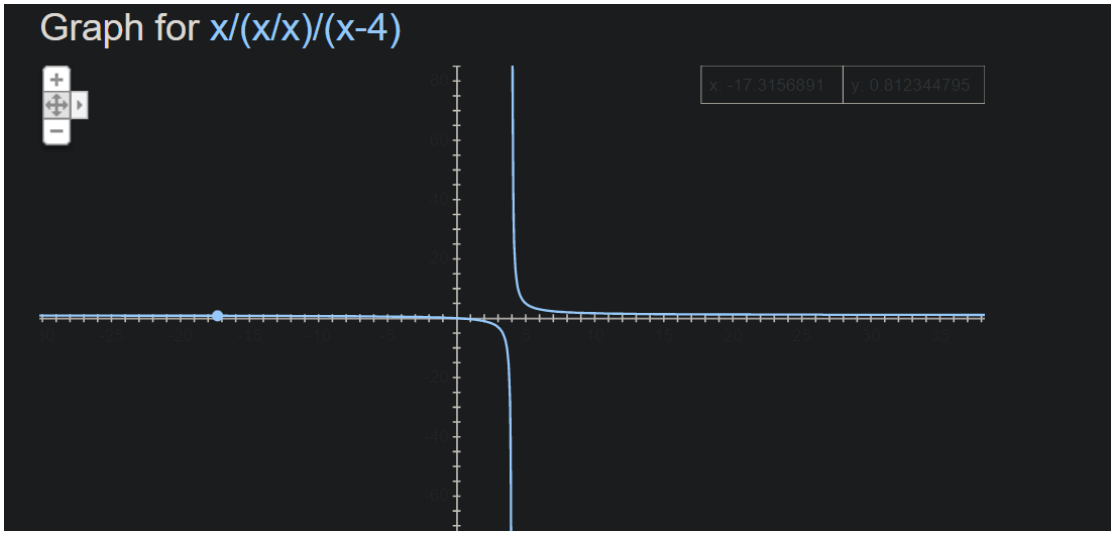
تابع شایستگی:با توجه به فرمول ام اس ای بدست اوردم
تولید جمعیت اولیه:به صورت
نحوه انتخاب والدین:ابتدا با استفاده از درصد های اولیه ، نسل خوب را
با هم کراس اور و سپس نسل خوب را با نسل بد کراس اور میکنیم و
نحوه تولید مثل:نسل اضافه شده را جداگانه اضافه کرده و در اخر به
همان اندازه ای که اول بود بهترین ها را نگه میداریم
نحوه ترکیب متقاطع و جهش: هنگام انتخاب نود برای ترکیب کردن ،
با احتمال کمی ممکن هست هویت گره ترکیب شده عوض شود
شرط خاتمه:هزار بار تولید مثل انجام میشود
چالش ها :طی گزارش نوشته شده

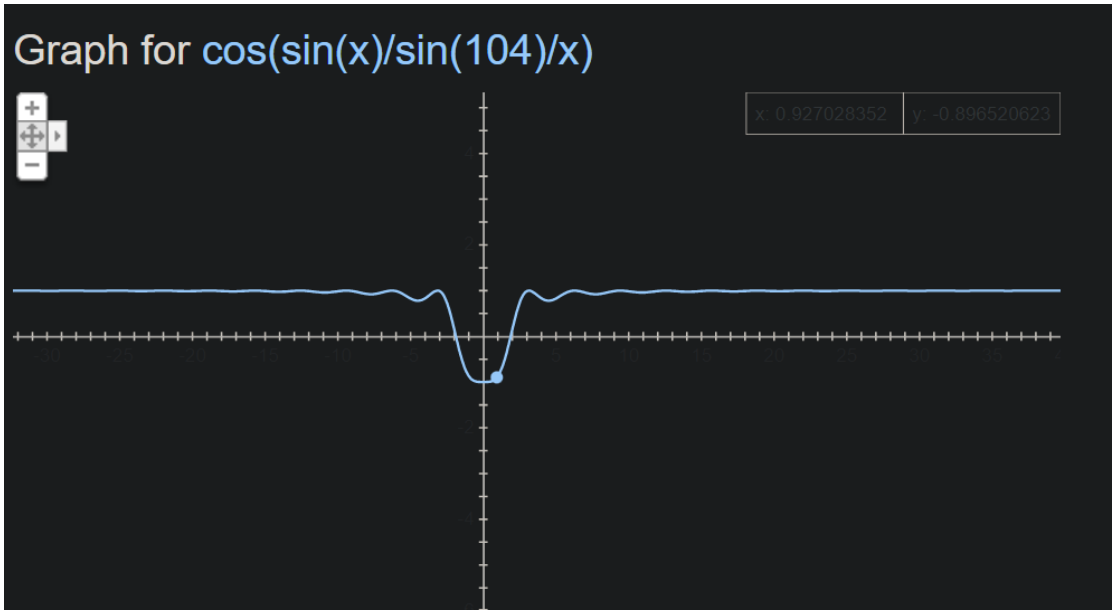
Target function:

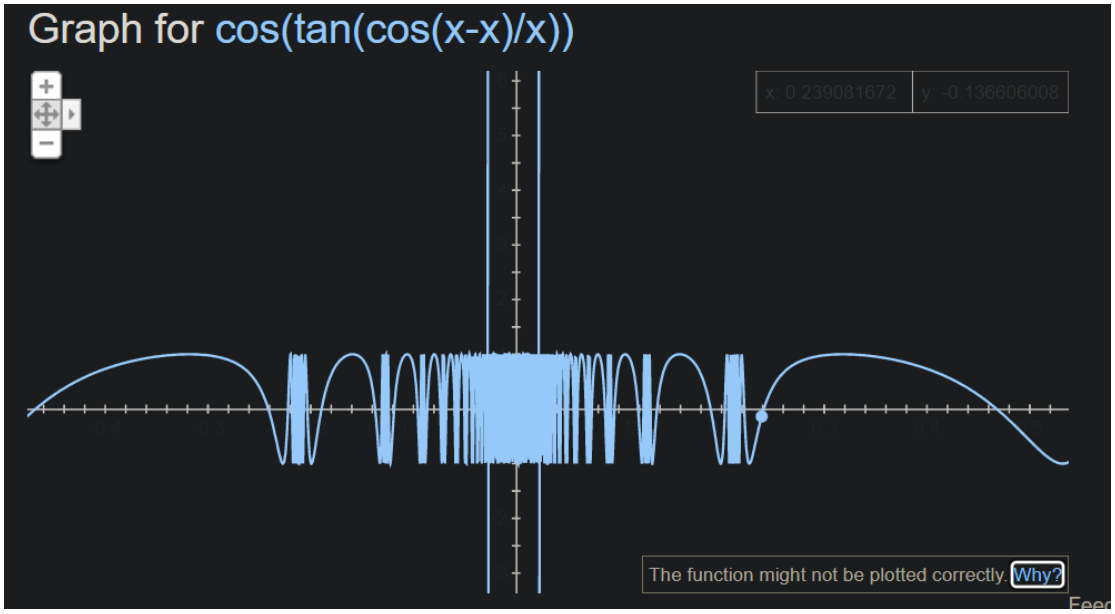
```
2 references | 2 references | 1 reference | 1 reference
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case = 80, maximum_variable_value = 300 , minumum_variable_value = -300 ;
14 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 30, worst_mse_percent = 50 ;//,best_mse_percent = 60 ;
2 references
const double expected_mse =0.00001 ;
```

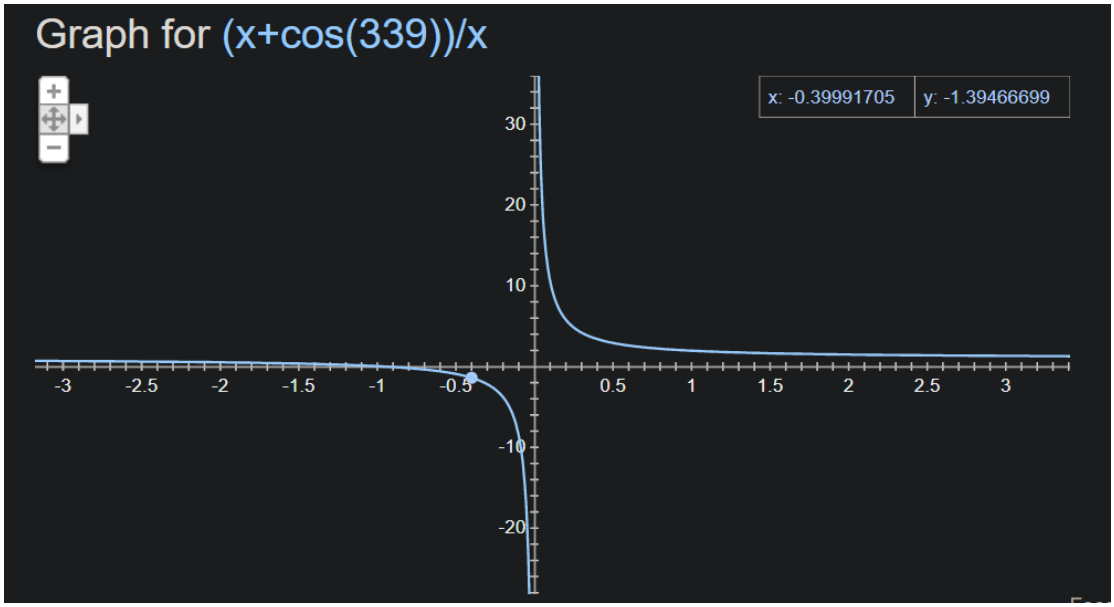


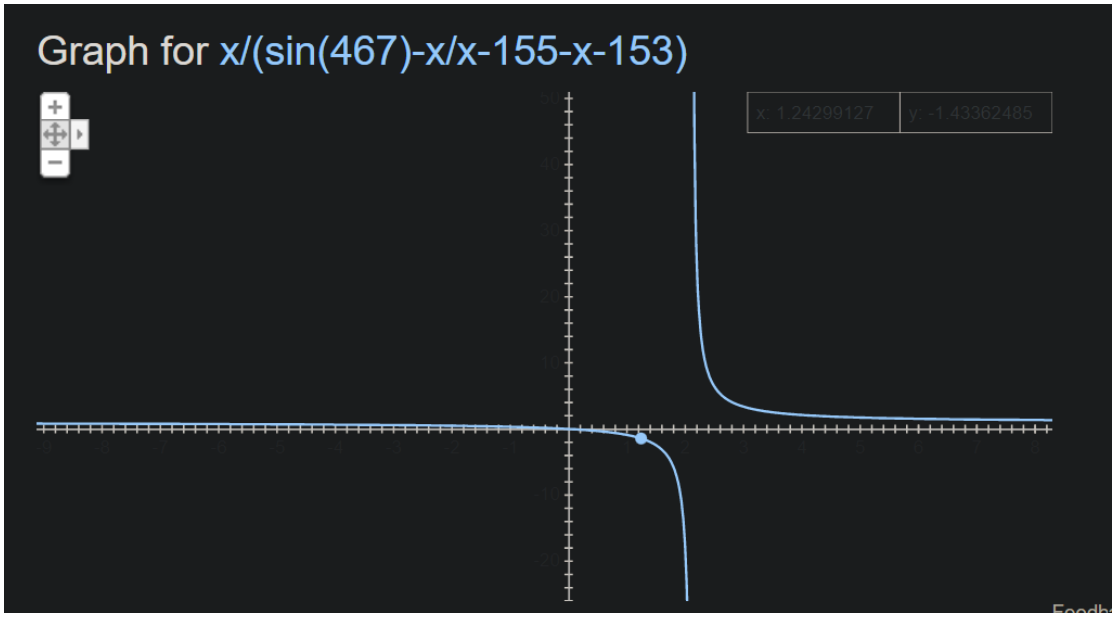
Result function:

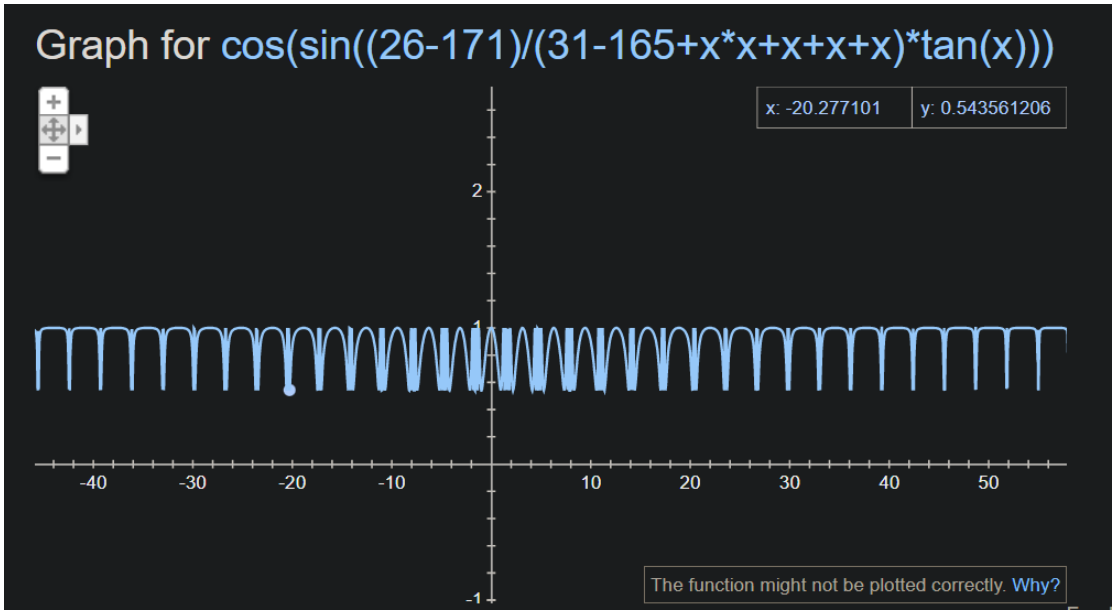



Graph for cos(sin(x)/sin(104)/x)
x: 0.927028352
y: -0.896520623


Graph for cos(tan(cos(x-x)/x))
x: 0.239081672
y: -0.136606008
The function might not be plotted correctly. Why?


Graph for (x+cos(339))/x
x: -0.39991705
y: -1.39466699
30
20
10
-10
-20
-3
-2.5
-2
-1.5
-1
-0.5
0.5
1
1.5
2
2.5
3

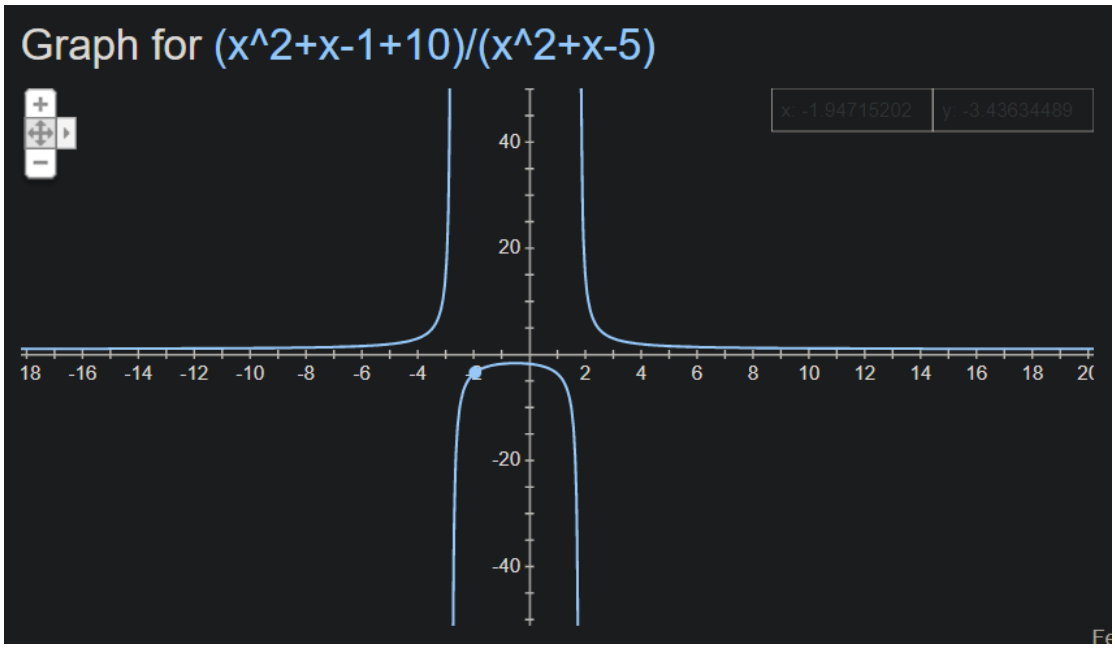

Graph for x/(sin(467)-x/x-155-x-153)
x: 1.24299127
y: -1.43362485



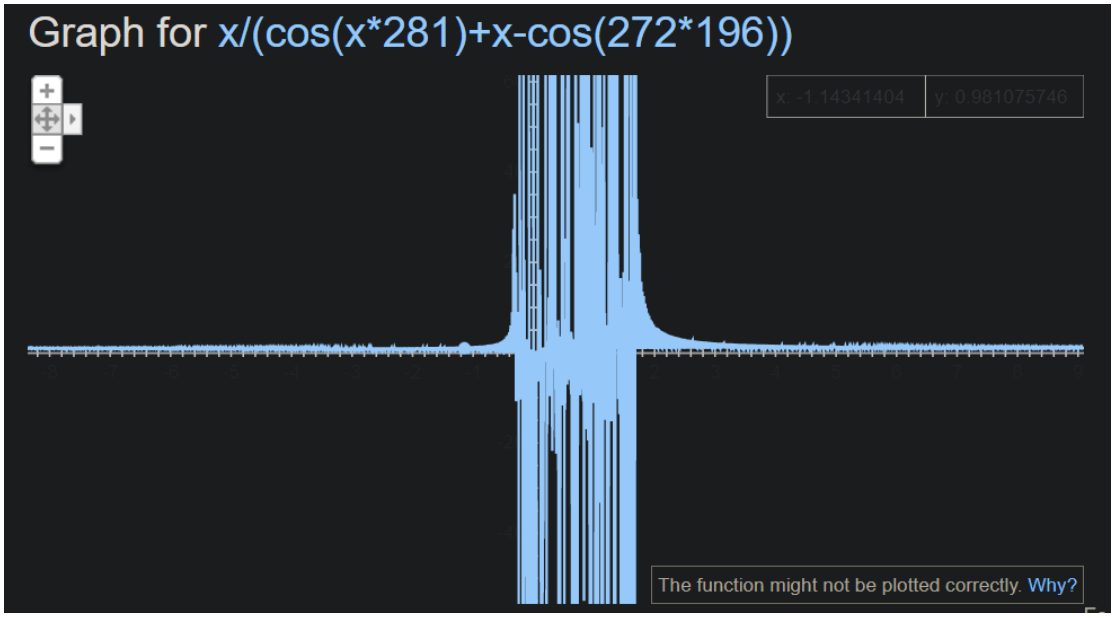
توابع قبلی را چک کردم و ان ها را مثل قبل به خوبی نمایش میداد ولی برای مشکل تابع بالا حل نشد پس تعداد تست هارا زیاد کردم و به صورت پیوسته تغییر دادم نه به صورت رندوم

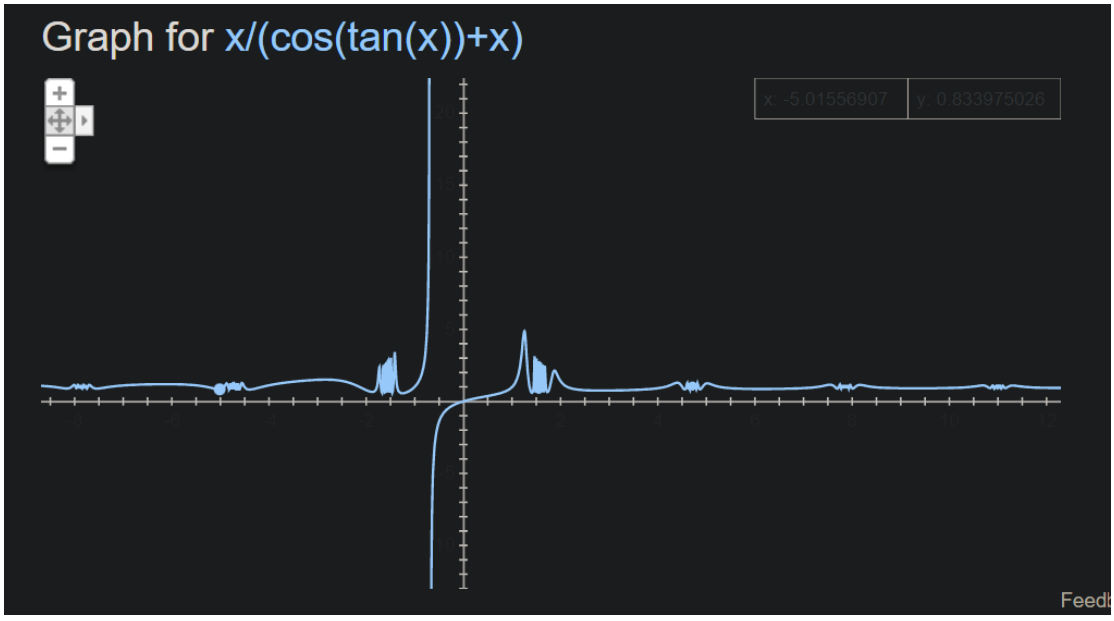
Target function:

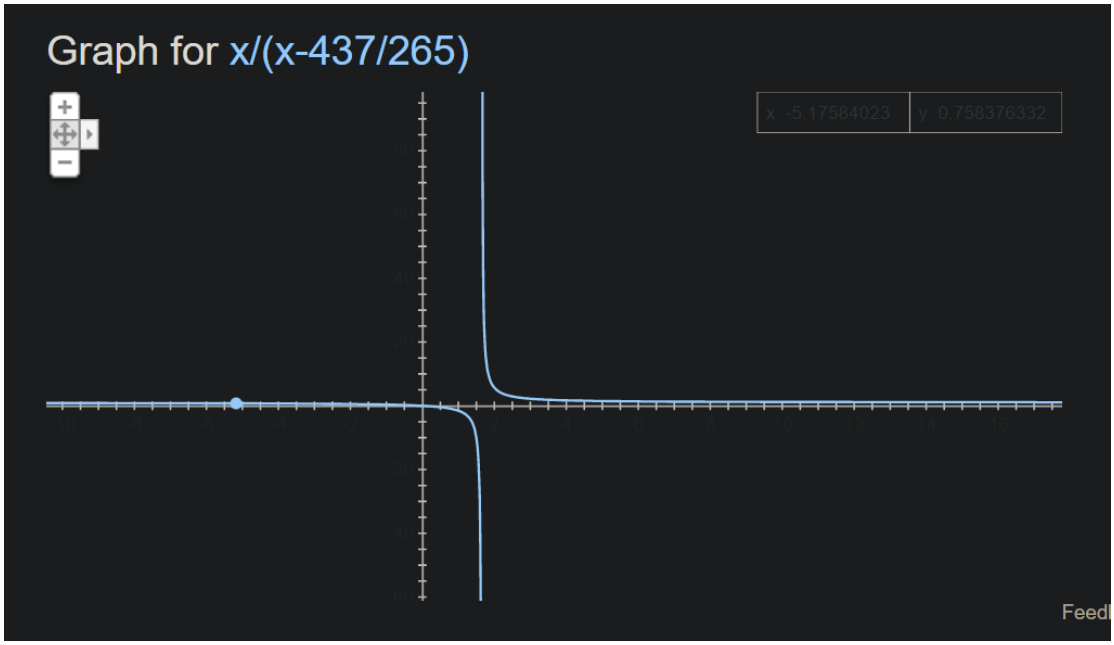
```
2 references | 2 references | 1 reference | 1 reference
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
4 references | 2 references | 2 references
const int test_case =200, maximum_variable_value = 100 , minumum_variable_value = -100 ;
14 references
static  int tree_instance_count = 500 ;
1 reference | 1 reference
const int last_generation_percent = 30, worst_mse_percent = 50 ;//,best_mse_percent = 60 ;
2 references
const double expected_mse =0.00001 ;
```



Result function:



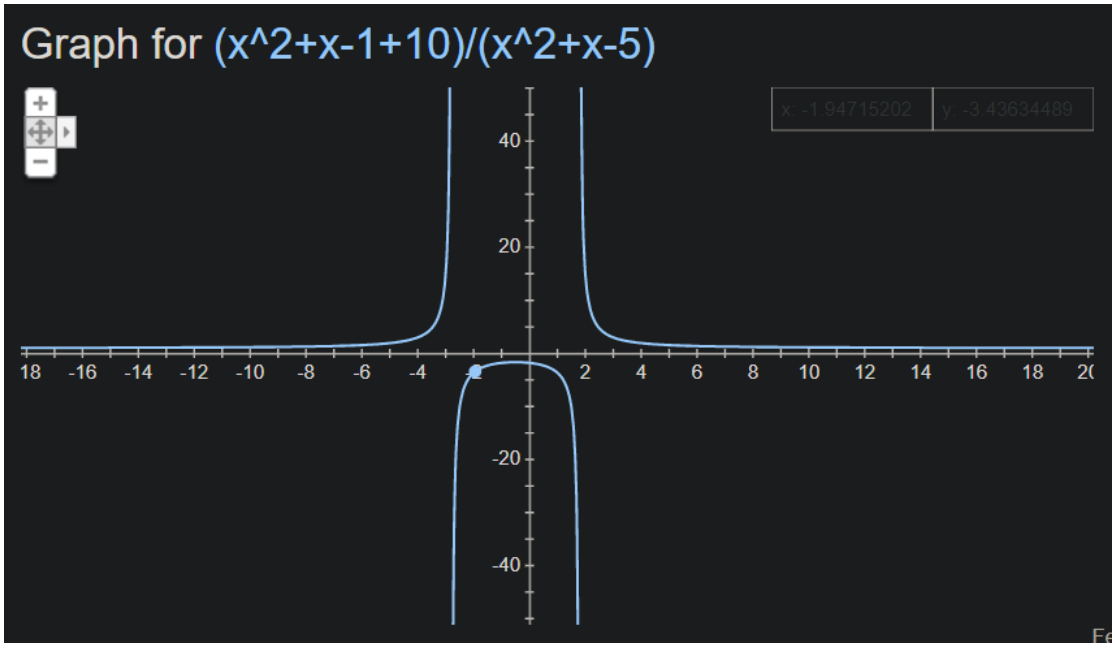




حالا تعداد درخت های اولیه را زیاد میکنیم

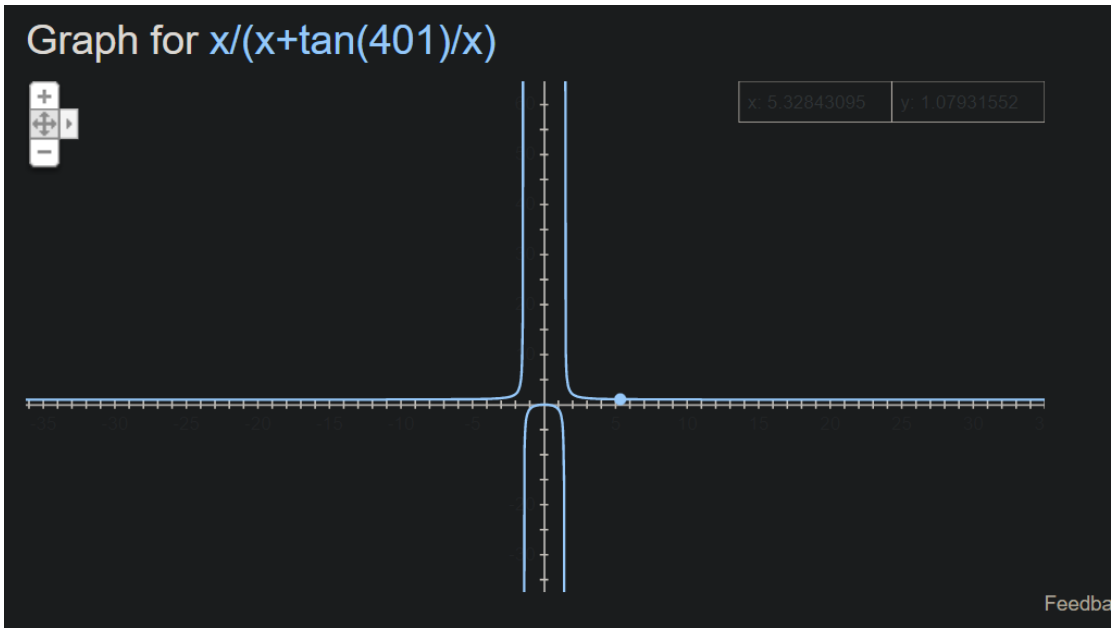
Target function:

```
const int limit_of_operator = 7 , limit_for_depth =15 , limit_for_constant = 500 , limit_for_type = 3 ;
const int test_case =200, maximum_variable_value = 100 , minumum_variable_value = -100 ;
static  int tree_instance_count = 1000 ;
const int last_generation_percent = 30, worst_mse_percent = 50 ;//,best_mse_percent = 60 ;
const double expected_mse =0.00001 ;
```

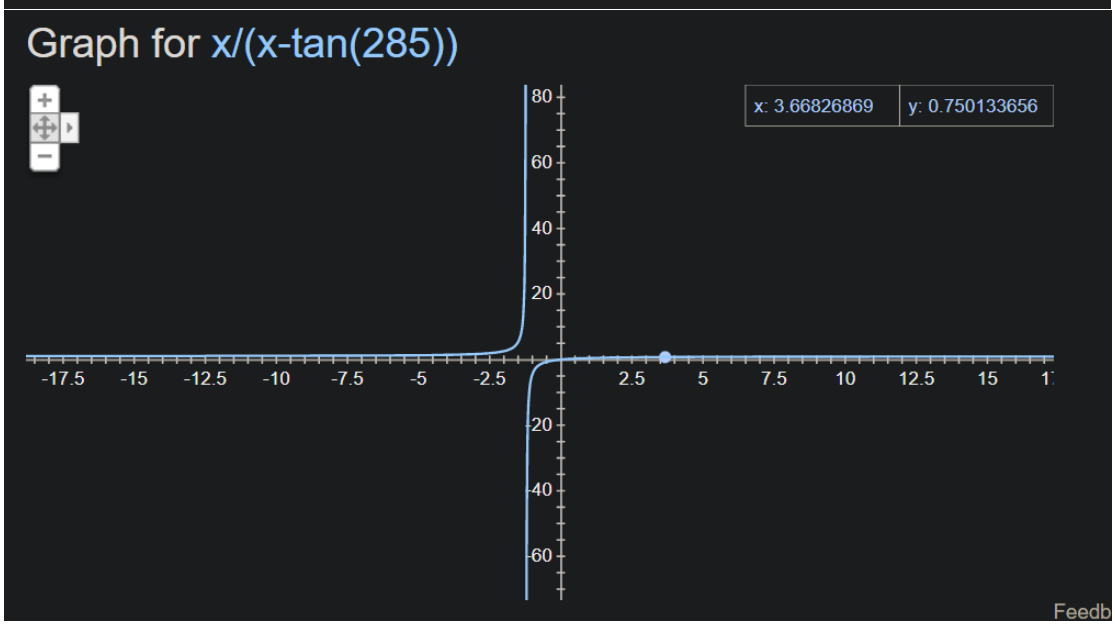


Result function:

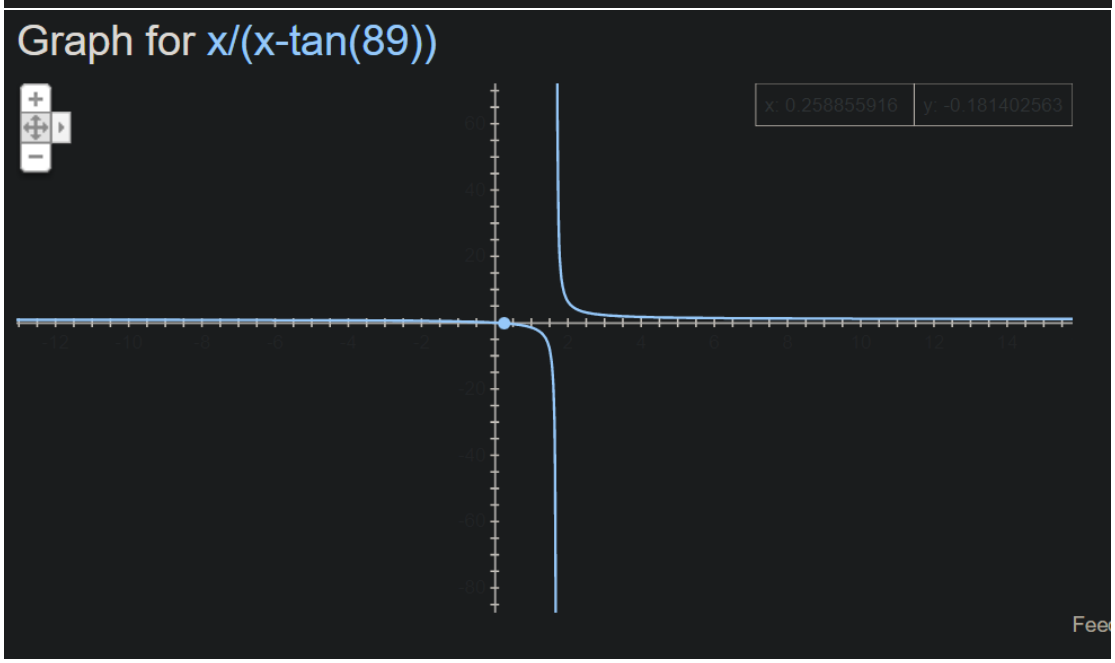
```
trees[0] with mse :2.1359163535623122 with time 00:02:22.8769536
(x/(x+(tan(401)/x)))
```




```
trees[0] with mse :1.9868045374362293 with time 00:03:29.2787409
(x/(x-tan(285)))
```



```
trees[0] with mse :1.020767929272296 with time 00:02:09.8221424
(x/(x-tan(89)))
```




```
trees[0] with mse :2.1813507098284535 with time 00:09:11.8273957
((x-tan((x/x)))/x)
```