

Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и информационных технологий
Кафедра вычислительной математики и программирования

Отчет по курсовому проекту
по курсу «Численные
Методы»

Студент: Иларионов Д.А.

Группа: М8О-308Б

Преподаватель: Сластушенский Ю.В.

Курсовая работа

1. Тема КР:

Решение нелинейных уравнений методом продолжения по параметру.

2. Вариант : 5

3. Алгоритм:

Когда реализация в разы сложнее самого алгоритма.. Данный алгоритм я сам не смог найти, когда я искал его, мне вылезало много непонятный сайтов с диффурами, ни одного теста или хотя бы шагов алгоритма. Я отчаялся и попросил преподавателя отправить какую-нибудь информацию для данной темы по КП. Он мне помог и кинул пару сайтов, которые я нигде не видел, но там все описано очень хорошо и подробно. На вход принимается функция, начальное приближение и интервал параметра. Решается уравнение методом Ньютона (из приближения вычитаем частное значения функции и значения производной). Оно решается так несколько раз. Каждый раз в виде приближения устанавливается предыдущее решение функции, а в качестве параметра – следующий с шагом. Например приближение у нас $x = 300$, а параметр с 1 до 5 с шагом 0.5. Мы решаем методом Ньютона с приближением 300 с параметром 1. Допустим, у нас вышло 324. Тогда мы решаем уже методом Ньютона с приближением 324 с параметром 1.5. Вышло 368 например, решаем с приближением 368 с параметром 2. И так далее, до того, как параметр не станет равным 5. Потом просто строим график по получившимся значениям. Довольно простой алгоритм, но на данную курсовую у меня ушло около недели работы. А дело не в алгоритме. А дело, в его реализации. (Хотя если бы я делал его в питоне, где уже есть библиотеки и функции работы с функциями, вычисления производной, то у меня ушло всего около дня, если не меньше. Но я не уверен, что так можно было бы, и мне казалось это слишком просто).

4. Среда разработки:

Adobe Animate CC – язык – JavaScript (Canvas)

Ссылка на проект на Github: https://denisolenison.github.io/numeric_methods_KP/

Ссылка на исх. код на Github: https://github.com/denisolenison/numeric_methods_KP

5. Реализация

Это сущий ужас. Вот серьезно. И дело не в реализации алгоритма самого. А сложное – это было создать интерфейс, структуру функций. А самое сложное – вычисление производной по этой функции! Казалось бы, что это просто долго. Но я так сильно уже запутался в собственных структурах, что очень часто возникали баги, например, что я не в ту структуру зашел, и очень много часов уходило на то, чтобы ловить их. Самое сложное – сделать нормальные выходы из функций после генерации из нее производной. Многие значения передаются по ссылке, из-за чего приходилось делать их копии. К счастью, в JavaScript вроде есть автоматический сборщик мусора, так что хоть с проблемами с освобождением памяти не пришлось сталкиваться, иначе, это был бы ад просто. Теперь о самой реализации. Есть структура формулы. И ссылка на всю формулу и на текущее место. В формуле могут быть объекты 6 типов: число, оператор, возведение в степень, корень, параметр и функция. У всех этих структур есть общий параметр `ist`, по которому можно понять, какая у нас структура. Число имеет тип, это может быть либо константа `pi` или число эйлера (для более красивого отображения), либо `number` и произвольное число. Число вводится в

отдельном поле, а потом мы его добавляем нажимая на N, либо выбираем функции, связанные с данным числом. Оператор простой тип, в котором просто в его типе указывается оператор символично. Возведение в степень возводит предыдущий элемент в определенную степень, аргументом является формула. Параметр и корень тоже простые типы. Функция – самый сложный тип. Во-первых, задается то, что внутри функции. Это ссылка на другую формулу. Когда мы выбираем функцию, то мы переходим в нее. Когда мы берем закрывающую скобку, выходим обратно. Скобки тоже представлены в виде функций. Также у некоторых функций есть еще аргумент – дополнительная формула, которая, например у логарифма определяет основание, либо у корня степень, либо у показательной функции число/параметр, которое мы возводим в степень. В то время, как мы возводим что-то в степень параметра или числа – является типом степени, то показательная функция, когда мы возводим само число или параметр в степень с формулой – является функцией. И из каждой формулы есть выход – ссылка на предыдущий массив. У начальной формулы он равен null. Например у нас функция $\cos(x + p) - \sin(p - x) = 0$. То в функции косинус у нас аргумента нет, формула примерно `formula([root, operator('+'), param])`, а выход – `formula([func(cos), operator('-'), func(sin)])`. На массивы указывает атрибут `inner`, на выход – `outer`. Более того, в функциях, `inner` указывает на формулу, и в формуле `inner` указывает на массив. Именно из-за этого и была основная путаница, я думаю, что данные атрибуты надо было назвать по-разному, потому что при копировании часто неправильно копировалось, например, вместо формулы массив и просто нужные атрибуты были равны null. И я очень долго сидел, чтобы понять проблему. Вот я и реализовал свою собственную программу построения функций. Дальше еще сложнее – производные. Мы разбиваем массив начальной формулы на части, между которыми стоят операторы '+' и '-'. Также в частях мы заменяем все операторы '/' на '*', а следующий объект возводим в -1 степень. Сокращаем степени в одну. У нас примерно получается массив [объект, степень, оператор(*), объект, степень, оператор(*), объект, степень]. Рекурсивно вычисляем производную произведения, как [объект, степень] и [объект, степень, оператор(*), объект, степень]. Если правая часть = [], то мы просто вычисляем производную левой части. Таким образом ответом будет (произв л.ч)*(пр.ч) + (произв пр.ч)*(л.ч). И так рекурсивно. Затем для следующей части, и так далее, пока не конец формулы. Для функций делаем так. (степень функции)*(функция^(ст-1))*(произв. функции с той же внутр. формулой)*(произв. формулы в функции). Последнее мы так же запускаем рекурсивно. Производная функции определяется в отдельной процедуре и зависит от типа функции. Для скобок она просто равна 1. Эта производная нужна нам для метода Ньютона, однако я сделал кнопку, которая просто преобразует формулу в производную в окошке. Но тут до сих пор некоторые проблемы с выходами при удалении поэлементно, лучше в таком образе формулу удалить целиком. Как это исправить – не знаю, но скорее всего решение окажется простым, потому что я уже сильно запутался в своих же структурах, а дебажить это довольно тяжело и занимает много времени. Также есть процедура сокращения. Например, $2+3$ преобразуется в 5. x^0 в 1. Это сделано для меньшей громоздкости формул, но это все еще работает не идеально. Например, производные в итоге получаются очень громоздкими, но зато правильными. Есть еще кое-что, но писать об этом я не буду в отчет)). Функция подставления значений в формулу работает так. Сначала подставляются X и P. Потом рекурсивно вычисляются функции. Затем возведение в степень. Затем умножение/деление. Затем сложение/вычитание. Все как в правилах порядка в математике. Далее уже пользователь выбирает начальное приближение, интервал параметра и количество продолжений. Нажимает на кнопку решить. Идет вычисление для первого приближения. Если значение получилось = NaN (обычно при комплексных числах), $+\infty$, $-\infty$, или метод Ньютона выполняется очень много раз, но разница между значениями постоянно большая, то это значит, что нет решений. И выводится сообщений, что решений для такого интервала параметра, приближения или просто формулы не найдено. Очень редко бывает это ошибочным, но почему-то часто зависит от начального приближение. Думаю, тут проблема уже в самом методе Ньютона. Если все ОК, то уже дальше идет алгоритм продолжения по параметру и выводится график (с осями параметра и значения корня X). График строится поточечно и соединяется линиями, количество точек зависит от количества продолжений. График можно в любое время закрыть и написать другую функцию или выбрать

другой интервал параметров/приближение, либо и то и то. Можно еще много чего написать, но я думаю, чтобы лучше понять реализацию, нужно посмотреть на сам код.

6. Руководство для пользователя

Число вводится в сером окошке слева. Оно не вводится сразу в формулу, а вводится в отдельные переменные. При нажатии на кнопку N оно преобразуется и добавляется в формулу. Также при выборе функции или степени, где нарисована буква N, функция добавляется с этим числом. Можно число умножить или разделить на 10. Кнопка C стирает 1 цифру или точку числа. Кнопка CE стирает число полностью. Чуть правее – операторы +, -, *, /. Их можно ставить только после числа, корня, параметра, степени или функции. Два оператора подряд ставить нельзя. Сверху кнопки N – добавляет число в формулу. P – добавляет параметр. X – добавляет корень. Открывающая и закрывающая скобки – для нужного приоритета операций. Слева еще кнопки e, pi. Они меняют число на эту константу и число больше не вводится. Чтобы снова можно было самому вводить число, нужно нажать на эту же кнопку еще раз. Справа – функции. Некоторые – с аргументом, который чаще всего представляет параметр, число, или формулу из них (типа p*x). Кнопка clear очищает полностью формулу. Кнопка Del удаляет один объект из формулы. Кнопка f'(x) преобразует формулу в ее производную и меняет ее. Что делает розовая кнопка – писать я сюда не буду)). Кнопка optimize сверху сокращает некоторые части формулы. Справа задаем начальное приближение. Все значение справа устанавливаются равными N, тому числу, которое мы вводили. Сначала задаем Pmax, а потом Pmin. Либо наоборот, если числа отрицательные. Но мы не можем задать Pmin > Pmax, либо Pmax < Pmin. Устанавливаем количество продолжений кнопками. Нажимаем решить – и опа, получаем ошибку. Ну или если все норм, то появляется новое окошко с графиком. Можем закрыть его и еще поэкспериментировать. Думаю, это все.

7. Тесты (скриншоты из программы с разными функциями):

Также, некоторые тесты содержатся в видео, по данной ссылке:

<https://www.youtube.com/watch?v=I6CsHvRKrmU>

Этап 1: Генерация уравнения

Уравнение (8 / 300)

Optimize

< >

$$F(X,P) = \sin(P \cdot X) = 0$$

Текущее число

N = 0

CLEAR

e

pi

C

Для генерации произвольного числа

7 8 9 C

4 5 6 CE

1 2 3 x10

- 0 , 10

N

P

X

)

^N

^(P+N)

log_{P+N}(

sqrt

^P

^(P-N)

e^(

sqrt[3]

^N

^(P*N)

log_{P-N}(

sqrt

P^(

^(P/N)

ln(

sqrt

Sin(

asin(

log_N(

f'(x)

cos(

acos(

log_P(

Del

tan(

atan(

log_P(

^w^

Начальное приближение X

~300

Set = N

P мин.

~-0.01

Set = N

P макс.

~-0.03

Set = N

Количество "продолжений"

15

+

-

ΔP (P2 - P1) =

~-0.00133

Решить!

Уравнение должно:

1) иметь корень

2) иметь параметр

3) иметь решение

4) не иметь незакрытых скобок, операторов в конце...

Курсовая работа по

Численным методам

Иларионов Денис (М80-308Б-17)

by denisolenison in June 2020

Этап 1: Генерация уравнения

Уравнение (8 / 300) Optimize < >

$F(X,P) =$

Те N = 0

График

Начальное приближение X

~300

Set = N

Р мин. ~0.01

Р макс. ~0.03

Set = N

Количество "продолжений" 15

ΔP (P2 - P1) = ~0.00133

Решить!

Уравнение должно:
1) иметь корень
2) иметь параметр
3) иметь решение
4) не иметь незакрытых скобок, операторов в конце...

Курсовая работа по
Численным методам
Иларионов Денис (М80-308Б-17)
by denisolenis in June 2020

Этап 1: Генерация уравнения

Уравнение (12 / 300) Optimize < >

$F(X,P) = \sin(P * X) - 1 = 0$

Те N = 0

Текущее число

CLEAR

Для генерации произвольного числа

7 8 9 C

4 5 6 CE

1 2 3 x10

- 0 , 10

+

-

x

÷

\wedge^N

$\wedge^{(P+N)}$

$\log_{P+N}()$

$\sqrt{\quad}$

\wedge^P

$\wedge^{(P-N)}$

e^{\quad}

$\sqrt[3]{\quad}$

$\wedge^{(P*N)}$

$\log_{P*N}()$

$\sqrt[N]{\quad}$

$\wedge^{(P/N)}$

$\ln()$

$\sqrt[P]{\quad}$

Sin(

asin(

$\log_N()$

$f'(x)$

cos(

acos(

$\log_P()$

DEL

tan(

atan(

$\log_p()$

\wedge^{\quad}

Начальное приближение X

~300

Set = N

Р мин. ~0.01

Р макс. ~0.05

Set = N

Количество "продолжений" 100

ΔP (P2 - P1) = ~0.0004

Решить!

Уравнение должно:
1) иметь корень
2) иметь параметр
3) иметь решение
4) не иметь незакрытых скобок, операторов в конце...

Курсовая работа по
Численным методам
Иларионов Денис (М80-308Б-17)
by denisolenis in June 2020

Этап 1: Генерация уравнения

Уравнение (12 / 300) Optimize < >

$F(X,P) =$

Те N = 0

Для e π N C 7 4 1 -

График

Начальное приближение X

Set = N

Р мин. Set = N Р макс. Set = N

Количество "продолжений" + -

$\Delta P (P2 - P1) =$

Решить!

Уравнение должно:
 1) иметь корень
 2) иметь параметр
 3) иметь решение
 4) не иметь незакрытых скобок, операторов в конце...

Курсовая работа по
 Численным методам
 Иларионов Денис (М80-308Б-17)
 by denisolenis in June 2020

Этап 1: Генерация уравнения

Уравнение (13 / 300) Optimize < >

$F(X,P) = \ln(P * X^2) - X = 0$

Текущее число N = 0 CLEAR

Для генерации произвольного числа

e π N P X $)$ \wedge^N $\wedge^{(P+N)}$ $\log_{P+N}()$ $\sqrt{\quad}$
 \wedge^P $\wedge^{(P-N)}$ e^{\quad} $\sqrt[3]{\quad}$
 N^{\quad} $\wedge^{(P \times N)}$ $\log_{P \times N}()$ $\sqrt[N]{\quad}$
 P^{\quad} $\wedge^{(P/N)}$ $\ln()$ $\sqrt[P]{\quad}$
 $\sin()$ $\arcsin()$ $\log_N()$ $f'(X)$
 $\cos()$ $\arccos()$ $\log_P()$ DEL
 $\tan()$ $\arctan()$ $\log_{P^{\quad}}()$ $\wedge^{\omega^{\quad}}$

Начальное приближение X

Set = N

Р мин. Set = N Р макс. Set = N

Количество "продолжений" + -

$\Delta P (P2 - P1) =$

Решить!

Уравнение должно:
 1) иметь корень
 2) иметь параметр
 3) иметь решение
 4) не иметь незакрытых скобок, операторов в конце...

Курсовая работа по
 Численным методам
 Иларионов Денис (М80-308Б-17)
 by denisolenis in June 2020

Этап 1: Генерация уравнения



8. Данный курсовой проект выполнен: 19 июня 2020.
9. Вывод (он будет коротким) – убил немало нервов, когда дебажил баги, связанные со структурами и выходами из формул. Зато, я укрепил некоторые навыки ООП, работу со структурами, ссылками, вложенными структурами, определенными видами деревьев, и рекурсивными функциями. Реализовал собственную программу представления функций, сделал сам вычисление производных любой функции. А также выполнил саму курсовую работу, реализовав алгоритм “продолжения по параметру” уже используя собственную программу и структуры. Я бы мог еще иногда обновлять данную программу, улучшая алгоритм сокращения функций, но пока не уверен, понадобится ли она еще где-нибудь, кроме данного курсового проекта. Зато еще укрепил некоторые навыки работы в среде разработки, стал более внимательным, и получил некоторый опыт, осознал, что лучше изначально представить структуры и классы простыми, реализовать несколько простых функций работы с ними и потом использовать уже их, чем постоянно переписывать одно и то же и путаться. Так правда удобнее, понадобится немного больше времени в начале, но зато потом уйдет намного меньше времени и будет меньше проблем. Полученный опыт мне еще пригодится, так как я и сам нередко работаю в данной среде разработки.

10. Код (JavaScript) – его очень много, около 3к строк. Тут содержится только тот код, который я писал сам. Код из файла index.js, который сгенерировался сам программой для создания интерфейса программы тут не учитывается.

```
1. //Число N
2. var custN = 0;
3. var custNafter = 0;
4. var hasPoint = false;
5. var vedZeros = 0;
6. var hasMinus = false;
7.
8. var isE = false;
9. var isPI = false;
10.
11.
12. //Формула
13. var formula = new theFormula([], null);
14.
15. //Текущее место
16. var curPlace = formula;
17.
18.
19. var graphOpen = false;
20.
21.
22. this.err2.visible = false;
23.
24.
25.
26. var xStar = 0;
27. var pMin = 0;
28. var pMax = 1;
29.
30. var conts = 10;
31.
32. var dP = 0.1;
33.
34.
35. var MAX_SIGNS = 32;
36. var fShift = 0;
37. var signsNow = 0;
38.
39.
40. window.setInterval(checkFps, 1000);
41.
42. function sound(src) {
43.     this.sound = document.createElement("audio");
44.     this.sound.src = src;
45.     this.sound.setAttribute("preload", "auto");
46.     this.sound.setAttribute("controls", "none");
47.     this.sound.style.display = "none";
48.     document.body.appendChild(this.sound);
49.     this.play = function () {
50.         this.sound.play();
51.     }
52.     this.pause = function () {
53.         this.sound.pause();
54.     }
55.     this.stop = function () {
56.         this.sound.pause();
57.         this.sound.currentTime = 0;
58.     }
59. }
60.
61. function checkFps() {
62.     fps = Math.max(1, fps2);
63.     fps2 = 0;
```

```

64. }
65.
66.
67. function theFormula(inner, outer) {
68.     this.inner = inner;
69.     this.outer = outer; //чтобы выйти из формулы
70. }
71.
72.
73. //+ - * /
74. function operator(type) {
75.     this.ist = "operator";
76.     this.type = type;
77. }
78.
79. //cos, sin, tg, ctg...
80. function funct(type, arg, outer) {
81.     this.ist = "function";
82.     this.type = type;
83.     this.inner = new theFormula([], outer);
84.     this.arg = arg;
85.     this.exit = false;
86. }
87.
88. function makeBracket(inner, outer) {
89.     var newBr = new funct("bracket", 0, outer);
90.     newBr.inner.inner = inner;
91.     newBr.exit = true;
92.     return newBr;
93. }
94.
95. function root() {
96.     this.ist = "root";
97. }
98.
99. function par() {
100.     this.ist = "param";
101. }
102.
103. //pi / e ...
104. function numb(typ, value) {
105.     this.ist = "numb";
106.     this.type = typ;
107.     this.value = value;
108. }
109.
110. //X^n - mun power, но n^(...) - mun function
111. function power(b, outer) {
112.     this.ist = "power";
113.     this.pow_to = new theFormula(b, outer);
114. }
115.
116.
117.
118. //копирование формулы
119. function copyF(form) {
120.     var newForm = new theFormula([], form.outer);
121.     var newInner = copyArr(form.inner, newForm);
122.     newForm.inner = newInner;
123.     return newForm;
124. }
125.
126.
127. function copyArr(inner, nOuter) {
128.     var newInner = [];
129.     for (var i = 0 ; i < inner.length ; ++i) {
130.         if (inner[i].ist == "operator") {
131.             newInner.push(new operator(inner[i].type));
132.         }

```

```

133.         else if (inner[i].ist == "root") {
134.             newInner.push(new root());
135.         }
136.         else if (inner[i].ist == "param") {
137.             newInner.push(new par());
138.         }
139.         else if (inner[i].ist == "numb") {
140.             newInner.push(new numb(inner[i].type, inner[i].value));
141.         }
142.         else if (inner[i].ist == "power") {
143.             newInner.push(new power(copyArr(inner[i].pow_to.inner), nOuter));
144.         }
145.         else if (inner[i].ist == "function") {
146.             newInner.push(copyFunc(inner[i], nOuter));
147.         }
148.     }
149.     return newInner;
150. }
151.
152.
153. function copyThis(thiss, nOuter) {
154.     var newThis;
155.     if (thiss.ist == "operator") {
156.         return new operator(thiss.type);
157.     }
158.     else if (thiss.ist == "root") {
159.         return new root();
160.     }
161.     else if (thiss.ist == "param") {
162.         return new par();
163.     }
164.     else if (thiss.ist == "numb") {
165.         return new numb(thiss.type, thiss.value);
166.     }
167.     else if (thiss.ist == "power") {
168.         return new power(copyArr(thiss.pow_to.inner), thiss.pow_to.outer);
169.     }
170.     else if (thiss.ist == "function") {
171.         return copyFunc(thiss, nOuter);
172.     }
173.     else {
174.         return thiss;
175.     }
176. }
177.
178.
179. function copyFunc(func1, newOuter) {
180.     var newFunc = new funct(func1.type, copyThis(func1.arg, newOuter), newOuter);
181.     newFunc.inner.inner = copyArr(func1.inner.inner, newFunc.inner);
182.     newFunc.exit = func1.exit;
183.     return newFunc;
184. }
185.
186.
187. //Добавляем цифру к числу
188. function addTo(num1, num2) {
189.     num1 *= 10;
190.     num1 += num2;
191.     num1 = Math.round(num1);
192.     return num1;
193. }
194.
195. function addX(X) {
196.     if ((custN.toString().length + custNafter.toString().length + vedZeros) <= 15) {
197.         if (hasPoint) {
198.             if (custNafter == 0 && X == 0) {
199.                 vedZeros += 1;
200.             }
201.             else {

```

```

202.         custNafter = addTo(custNafter, X);
203.     }
204. }
205.     else {
206.         custN = addTo(custN, X);
207.     }
208. }
209. }
210.
211. function setVZ(num) {
212.     var curSt = "";
213.     for (var i = 0 ; i < num ; ++i) {
214.         curSt += "0";
215.     }
216.     return curSt;
217. }
218.
219. //Количество ведущих нулей после 1 знака части после запятой
220. function zal(num) {
221.     var nStr = num.toString();
222.     var zeros = 0;
223.     for (var i = 1 ; i < nStr.length ; ++i) {
224.         if (nStr[i] == "0") {
225.             zeros++;
226.         }
227.         else {
228.             break;
229.         }
230.     }
231.     return zeros;
232. }
233.
234. function wrOP(str) {
235.     if (str == "+" || str == "-") {
236.         return " " + str + " ";
237.     }
238.     else return str;
239. }
240.
241.
242. function toTheNum(first, second, zeros, sign) {
243.     var num = first;
244.     if (second != 0) {
245.         var logNum = Math.floor(Math.log(second) / Math.log(10)) + 1 + zeros;
246.         var num2 = second / Math.pow(10, logNum);
247.         return (num + num2) * sign;
248.     }
249.     else return num * sign;
250. }
251.
252.
253.
254.
255. function writeEq(formula) {
256.     var str = "";
257.     var elements = formula.inner.length;
258.     for (var i = 0 ; i < elements ; ++i) {
259.         str += writeThe(formula.inner[i]);
260.     }
261.     return str;
262. }
263.
264.
265. function writeThe(object) {
266.     if (object.ist == "operator") {
267.         return wrOP(object.type);
268.     }
269.     else if (object.ist == "root") {
270.         return "X";

```

```

271.     }
272.     else if (object.ist == "param") {
273.         return "P";
274.     }
275.     else if (object.ist == "numb") {
276.         if (object.type == "e") {
277.             return "e";
278.         }
279.         else if (object.type == "pi") {
280.             return "π";
281.         }
282.         else {
283.             return object.value;
284.         }
285.     }
286.     else if (object.ist == "power") {
287.         if (object.pow_to.inner.length == 1) {
288.             return "^" + writeThe(object.pow_to.inner[0]);
289.         }
290.         else {
291.             return "(" + writeEq(object.pow_to) + ")";
292.         }
293.     }
294.     else if (object.ist == "function") {
295.         var theTx = "";
296.         if (object.type == "power") {
297.             theTx += writeThe(object.arg) + "(" + writeEq(object.inner);
298.         }
299.         else if (object.type == "log") {
300.             theTx += "log" + "[" + writeEq(object.arg.inner) + "]" + "(" + writeEq(object.inner);
301.         }
302.         else if (object.type == "root") {
303.             theTx += "root" + "[" + writeEq(object.arg.inner) + "]" + "(" + writeEq(object.inner);
304.         }
305.         else if (object.type == "bracket") {
306.             theTx += "(" + writeEq(object.inner);
307.         }
308.         else {
309.             theTx += object.type + "(" + writeEq(object.inner);
310.         }
311.
312.         if (object.exit) {
313.             theTx += ")";
314.         }
315.         return theTx;
316.     }
317. }
318.
319.
320. function getThePart(text, begin, end) {
321.     var textAns = "";
322.     for (var i = begin ; i < end ; ++i) {
323.         textAns += text[i];
324.     }
325.     return textAns;
326. }
327.
328.
329.
330.
331. this.addEventListener("tick", main.bind(this));
332. function main() {
333.     if (isE) {
334.         this.custNum.text = "e";
335.     }
336.     else if (isPI) {
337.         this.custNum.text = "π";
338.     }
339.     else {

```

```

340.         if (hasPoint == false) {
341.             this.custNum.text = (hasMinus ? "-" : "") + Math.round(custN);
342.         }
343.         else {
344.             this.custNum.text = (hasMinus ? "-"
" : "") + Math.round(custN) + "." + setVZ(vedZeros) + (custNafter ? Math.round(custNafter) : "");
345.         }
346.     }
347.
348.     signsNow = writeEq(formula).length;
349.     this.curEq.text = getThePart(writeEq(formula), fShift, Math.min(fShift+32, signsNow));
350.     if (signsNow > fShift+32) {
351.         this.curEq.text += "..";
352.     }
353.
354.
355.     this.sizeTT.text = "(" + Math.round(signsNow) + " / 300)";
356.
357.     if (this.err2.visible) {
358.         this.err2.alpha -= 1/(5*fps);
359.     }
360.     if (this.err2.alpha <= 0) {
361.         this.err2.visible = false;
362.         this.err2.alpha = 1;
363.     }
364.
365.
366.     if (signsNow >= 300) {
367.         this.sizeTT.color = '#AA0000';
368.         this.warning.visible = true;
369.     }
370.     else {
371.         this.sizeTT.color = '#004400';
372.         this.warning.visible = false;
373.     }
374.
375.
376.     dP = (pMax - pMin) / (conts);
377.
378.     this.xStarT.text = "~" + Math.round(xStar*10000)/10000;
379.
380.     this.pMinT.text = "~" + Math.round(pMin*10000)/10000;
381.     this.pMaxT.text = "~" + Math.round(pMax*10000)/10000;
382.
383.
384.     this.continos.text = Math.round(conts);
385.     this.deltaP.text = "~" + Math.round(dP*100000)/100000;
386.
387. }
388.
389.
390. this.add1.addEventListener("click", add1f.bind(this));
391. function add1f() {
392.     addX(1);
393. }
394.
395. this.add2.addEventListener("click", add2f.bind(this));
396. function add2f() {
397.     addX(2);
398. }
399.
400. this.add3.addEventListener("click", add3f.bind(this));
401. function add3f() {
402.     addX(3);
403. }
404.
405. this.add4.addEventListener("click", add4f.bind(this));
406. function add4f() {
407.     addX(4);

```

```

408.     }
409.
410.
411.     this.add5.addEventListener("click", add5f.bind(this));
412.     function add5f() {
413.         addX(5);
414.     }
415.
416.     this.add6.addEventListener("click", add6f.bind(this));
417.     function add6f() {
418.         addX(6);
419.     }
420.
421.     this.add7.addEventListener("click", add7f.bind(this));
422.     function add7f() {
423.         addX(7);
424.     }
425.
426.     this.add8.addEventListener("click", add8f.bind(this));
427.     function add8f() {
428.         addX(8);
429.     }
430.
431.     this.add9.addEventListener("click", add9f.bind(this));
432.     function add9f() {
433.         addX(9);
434.     }
435.
436.     this.add0.addEventListener("click", add0f.bind(this));
437.     function add0f() {
438.         addX(0);
439.     }
440.
441.     var trig = 0;
442.     var trigTim = 0;
443.
444.     this.formuLeft.addEventListener("mouseover", setTrig.bind(this));
445.     function setTrig() {
446.         trig = -1;
447.     }
448.
449.     this.formuLeft.addEventListener("mouseout", setTrig2.bind(this));
450.     function setTrig2() {
451.         trig = 0;
452.         trigTim = 0;
453.     }
454.
455.     this.formuLeft.addEventListener("click", AsetTrig2.bind(this));
456.     function AsetTrig2() {
457.
458.         if (fShift > 0) {
459.             fShift -= 1;
460.         }
461.         trig = 0;
462.         trigTim = 0;
463.     }
464.
465.     this.formuRight.addEventListener("mouseover", setTrig12.bind(this));
466.     function setTrig12() {
467.         trig = 1;
468.     }
469.
470.     this.formuRight.addEventListener("mouseout", setTrig22.bind(this));
471.     function setTrig22() {
472.         trig = 0;
473.         trigTim = 0;
474.     }
475.
476.     this.formuRight.addEventListener("click", AsetTrig22.bind(this));

```

```

477.     function AsetTrig22() {
478.         if (signsNow > fShift+32) {
479.             fShift += 1;
480.         }
481.         trig = 0;
482.         trigTim = 0;
483.     }
484.
485.     var fpSSS = 0;
486.
487.     this.addEventListener("tick", settingTrigger.bind(this));
488.     function settingTrigger() {
489.         fpSSS += 1;
490.         if (trig == 1) {
491.             trigTim += 1/fps;
492.         }
493.         else if (trig == -1) {
494.             trigTim -= 1/fps;
495.         }
496.         if (trigTim >= 3) {
497.             if (signsNow > fShift+32 && fpSSS%3==0) {
498.                 fShift += 1;
499.             }
500.         }
501.         else if (trigTim <= -3 && fpSSS%3==0) {
502.             if (fShift > 0) {
503.                 fShift -= 1;
504.             }
505.         }
506.     }
507.
508.
509.     this.addPt.addEventListener("click", addPtF.bind(this));
510.     function addPtF() {
511.         hasPoint = true;
512.     }
513.
514.     this.removeOne.addEventListener("click", removeOneF.bind(this));
515.     function removeOneF() {
516.         if (vedZeros > 0 && custNafter==0) {
517.             vedZeros -= 1;
518.         }
519.         else if (!hasPoint) {
520.             custN = Math.floor(custN/10);
521.         }
522.         else if (custNafter != 0) {
523.             custNafter = Math.floor(custNafter/10);
524.         }
525.         else {
526.             hasPoint = false;
527.         }
528.     }
529.
530.     this.removeAll.addEventListener("click", removeAllF.bind(this));
531.     function removeAllF() {
532.         custN = 0;
533.         custNafter = 0;
534.         hasPoint = false;
535.         vedZeros = 0;
536.         hasMinus = false;
537.     }
538.
539.     this.mult10.addEventListener("click", mult10F.bind(this));
540.     function mult10F() {
541.         if ((custN.toString().length + custNafter.toString().length + vedZeros) <= 15) {
542.             if (!hasPoint || custNafter == 0 && vedZeros == 0) {
543.                 custN = Math.round(custN*10);
544.             }
545.             else if (vedZeros > 0) {

```



```

546.         custN = Math.round(custN*10);
547.         vedZeros -= 1;
548.     }
549.     else if (custNafter != 0) {
550.         var maxPtPt = Math.log(custNafter) / Math.log(10);
551.         var maxPtPt = Math.floor(maxPtPt);
552.         vedZeros += za1(custNafter);
553.         var t10power = Math.pow(10, maxPtPt);
554.         var valX = Math.floor(custNafter/t10power);
555.         custNafter = Math.round(custNafter - t10power*valX);
556.         custN = addTo(custN, valX);
557.     }
558.     if (custNafter == 0 && vedZeros == 0) {
559.         hasPoint = false;
560.     }
561. }
562. }
563.
564. this.div10.addEventListener("click", div10f.bind(this));
565. function div10f() {
566.     if ((custN.toString().length + custNafter.toString().length + vedZeros) <= 15) {
567.         var lastDigit = custN%10;
568.         if (lastDigit == 0) {
569.             if (hasPoint) {
570.                 vedZeros += 1;
571.             }
572.             custN = Math.floor(custN/10);
573.         }
574.         else if (custN !=0) {
575.             hasPoint = true;
576.             custN = Math.floor(custN/10);
577.             var newNum = lastDigit * Math.pow(10, vedZeros);
578.             vedZeros = 0;
579.             var newLog;
580.             if (custNafter != 0) {
581.                 newLog = Math.floor(Math.log(custNafter) / Math.log(10))+1;
582.             }
583.             else newLog = 0;
584.             custNafter += newNum * Math.pow(10, newLog);
585.         }
586.     }
587. }
588.
589. this.setE.addEventListener("click", setEf.bind(this));
590. function setEf() {
591.     if (!isE) {
592.         isE = true;
593.         isPI = false;
594.     }
595.     else {
596.         isE = false;
597.         isPI = false;
598.     }
599. }
600.
601. this.setPI.addEventListener("click", setPIf.bind(this));
602. function setPIf() {
603.     if (!isPI) {
604.         isPI = true;
605.         isE = false;
606.     }
607.     else {
608.         isPI = false;
609.         isE = false;
610.     }
611. }
612.
613.
614. this.senNxStar.addEventListener("click", senNxStarf.bind(this));

```

```

615.     function senNxStarf() {
616.         if (isE) {
617.             xStar = 2.718281828459;
618.             removeAllf();
619.         }
620.         else if (isPI) {
621.             xStar = 3.14159265359;
622.             removeAllf();
623.         }
624.         else {
625.             xStar = toTheNum(custN, custNafter, vedZeros, bll(hasMinus));
626.             removeAllf();
627.         }
628.     }
629.
630.
631.     this.senPMin.addEventListener("click", senPMinf.bind(this));
632.     function senPMinf() {
633.         if (isE) {
634.             if (2.718281828459 < pMax) {
635.                 pMin = 2.718281828459;
636.                 removeAllf();
637.             }
638.         }
639.         else if (isPI) {
640.             if (3.14159265359 < pMax) {
641.                 pMin = 3.14159265359;
642.                 removeAllf();
643.             }
644.         }
645.         else {
646.             if (toTheNum(custN, custNafter, vedZeros, bll(hasMinus)) < pMax) {
647.                 pMin = toTheNum(custN, custNafter, vedZeros, bll(hasMinus));
648.                 removeAllf();
649.             }
650.         }
651.     }
652.
653.
654.     this.senPMax.addEventListener("click", senPMaxf.bind(this));
655.     function senPMaxf() {
656.         if (isE) {
657.             if (2.718281828459 > pMin) {
658.                 pMax = 2.718281828459;
659.                 removeAllf();
660.             }
661.         }
662.         else if (isPI) {
663.             if (3.14159265359 > pMin) {
664.                 pMax = 3.14159265359;
665.                 removeAllf();
666.             }
667.         }
668.         else {
669.             if (toTheNum(custN, custNafter, vedZeros, bll(hasMinus)) > pMin) {
670.                 pMax = toTheNum(custN, custNafter, vedZeros, bll(hasMinus));
671.                 removeAllf();
672.             }
673.         }
674.     }
675.
676.
677.     var trig2 = 0;
678.     var trigTim2 = 0;
679.     var fpSSS2 = 0;
680.
681.     this.addEventListener("tick", settingTrigger2.bind(this));
682.     function settingTrigger2() {
683.         fpSSS2 += 1;

```

```

684.         if (trig2 == 1) {
685.             trigTim2 += 1/fps;
686.         }
687.         else if (trig2 == -1) {
688.             trigTim2 -= 1/fps;
689.         }
690.         if (trigTim2 >= 3) {
691.             if (conts < 300 && fpSSS2%3<=1) {
692.                 conts += 1;
693.             }
694.         }
695.         else if (trigTim2 <= -3 && fpSSS2%3<=1) {
696.             if (conts > 1) {
697.                 conts -= 1;
698.             }
699.         }
700.
701.         if (fShift > signsNow-32) {
702.             fShift = Math.max(0, signsNow-32);
703.         }
704.
705.
706.     }
707.
708.     this.plusCon.addEventListener("click", plusConf.bind(this));
709.     function plusConf() {
710.         if (conts < 300) {
711.             conts++;
712.         }
713.         trigTim2 = 0;
714.         trig2 = 0;
715.     }
716.
717.     this.plusCon.addEventListener("mouseover", plusConf2.bind(this));
718.     function plusConf2() {
719.         if (conts < 300) {
720.             trig2 = 1;
721.         }
722.     }
723.
724.     this.plusCon.addEventListener("mouseout", plusConf3.bind(this));
725.     function plusConf3() {
726.         trigTim2 = 0;
727.         trig2 = 0;
728.     }
729.
730.
731.     this.minusCon.addEventListener("click", minusConf.bind(this));
732.     function minusConf() {
733.         if (conts > 1) {
734.             conts--;
735.         }
736.         trigTim2 = 0;
737.         trig2 = 0;
738.     }
739.
740.     this.minusCon.addEventListener("mouseover", minusConf2.bind(this));
741.     function minusConf2() {
742.         if (conts > 1) {
743.             trig2 = -1;
744.         }
745.     }
746.
747.     this.minusCon.addEventListener("mouseout", minusConf3.bind(this));
748.     function minusConf3() {
749.         trigTim2 = 0;
750.         trig2 = 0;
751.     }
752.

```

```

753.
754. this.graphWindow.closGraph.addEventListener("click", closGR.bind(this));
755. function closGR() {
756.     this.graphWindow.visible = false;
757.     graphOpen = false;
758. }
759.
760.
761. function badNumber(num) {
762.     if (isNaN(num) || num == Infinity || num == -Infinity) {
763.         return true;
764.     }
765.     else {
766.         return false;
767.     }
768. }
769.
770.
771. var pArray = [];
772. var xArray = [];
773.
774.
775. function newTone_Method(x, p) {
776.     var xPrev = x;
777.     var xNext = x;
778.     var iters = 0;
779.
780.     while ((Math.abs((xNext/xPrev) - 1) > 0.0000001 && iters < 1000) || iters < 5) {
781.
782.         var forDer = findDerivative(copyF(formula), formula.outer);
783.         for (var i = 0 ; i < 10 ; ++i) {
784.             forDer.inner = optimize(forDer.inner, 0);
785.         }
786.
787.         var fxn = podstava(xNext, p, copyF(formula));
788.         var dfxn = podstava(xNext, p, copyF(forDer));
789.         xPrev = xNext;
790.         xNext = xPrev - (fxn/dfxn);
791.         ++iters;
792.     }
793.     if (iters >= 999) {
794.         return Infinity;
795.     }
796.     else {
797.         return xNext;
798.     }
799. }
800.
801.
802.
803. this.solveBt.addEventListener("click", solving.bind(this));
804. function solving() {
805.     pArray = [];
806.     xArray = [];
807.
808.     var xPrev = xStar;
809.     var xNext = xStar;
810.     var iters = 0;
811.
812.     while ((Math.abs((xNext/xPrev) - 1) > 1e-15 && iters < 1000) || iters < 5) {
813.
814.         for (var i = 0 ; i < 10 ; ++i) {
815.             formula.inner = optimize(formula.inner, 0);
816.         }
817.         var forDer = findDerivative(copyF(formula), formula.outer);
818.         for (var i = 0 ; i < 10 ; ++i) {
819.             forDer.inner = optimize(forDer.inner, 0);
820.         }
821.

```

```

822.         var fxn = podstava(xNext, pMin, copyF(formula));
823.         //console.log(fxn);
824.         var dfxn = podstava(xNext, pMin, copyF(forDer));
825.         xPrev = xNext;
826.         xNext = xPrev - (fxn/dfxn);
827.         ++iters;
828.     }
829.
830.     if (iters > 999 || badNumber(xNext)) {
831.         this.err2.visible = true;
832.         this.err2.alpha = 1;
833.     }
834.     else {
835.         pArray.push(pMin);
836.         xArray.push(xNext);
837.         var step = 1;
838.         var curP = pMin + dP;
839.         var xThis = xNext;
840.         while (Math.round(step) <= Math.round(conds)) {
841.             xThis = newTone_Method(xThis, curP);
842.             if (badNumber(xThis)) {
843.                 break;
844.             }
845.             else {
846.                 pArray.push(curP);
847.                 xArray.push(xThis);
848.                 ++step;
849.                 curP = curP + dP;
850.             }
851.         }
852.
853.         stage.addChild(this.graphWindow);
854.         this.graphWindow.visible = true;
855.         this.graphWindow.x = 500;
856.         this.graphWindow.y = 370;
857.
858.
859.         graphOpen = true;
860.
861.         console.log(pArray);
862.         console.log(xArray);
863.
864.         var pMinW = 500 + this.graphWindow.xyMin.x;
865.         var xMinW = 370 + this.graphWindow.xyMin.y;
866.         var pMaxW = 500 + this.graphWindow.xMax.x;
867.         var xMaxW = 370 + this.graphWindow.yMax.y;
868.
869.         var xMin = xArray[0];
870.         var xMax = xArray[0];
871.
872.
873.         for (var i = 0 ; i < xArray.length ; ++i) {
874.             if (xArray[i] < xMin) {
875.                 xMin = xArray[i];
876.             }
877.             if (xArray[i] > xMax) {
878.                 xMax = xArray[i];
879.             }
880.         }
881.
882.         var dist = Math.abs(xMax - xMin);
883.         xMax += dist*0.15;
884.
885.         if (dist == 0) {
886.             xMax += 0.005;
887.             xMin -= 0.005;
888.         }
889.         else if (dist < 0.004) {
890.             xMax += 0.002;

```

```

891.         xMin -= 0.002;
892.     }
893.
894.     this.graphWindow.y1.text = Math.round(xMin*1000)/1000 + "";
895.
896.     this.graphWindow.y2.text = Math.round((xMin + (xMax - xMin)*(1/6))*1000)/1000 + "";
897.     this.graphWindow.y3.text = Math.round((xMin + (xMax - xMin)*(2/6))*1000)/1000 + "";
898.     this.graphWindow.y4.text = Math.round((xMin + (xMax - xMin)*(3/6))*1000)/1000 + "";
899.     this.graphWindow.y5.text = Math.round((xMin + (xMax - xMin)*(4/6))*1000)/1000 + "";
900.     this.graphWindow.y6.text = Math.round((xMin + (xMax - xMin)*(5/6))*1000)/1000 + "";
901.
902.     this.graphWindow.y7.text = Math.round(xMax*1000)/1000 + "";
903.
904.     this.graphWindow.x1.text = Math.round(pMin*1000)/1000 + "";
905.
906.     this.graphWindow.x2.text = Math.round((pMin + (pMax - pMin)*(1/5))*1000)/1000 + "";
907.     this.graphWindow.x3.text = Math.round((pMin + (pMax - pMin)*(2/5))*1000)/1000 + "";
908.     this.graphWindow.x4.text = Math.round((pMin + (pMax - pMin)*(3/5))*1000)/1000 + "";
909.     this.graphWindow.x5.text = Math.round((pMin + (pMax - pMin)*(4/5))*1000)/1000 + "";
910.
911.     this.graphWindow.x6.text = Math.round(pMax*1000)/1000 + "";
912.
913.
914.     for (var i = 0 ; i < xArray.length ; ++i) {
915.         var point = new lib.bigPoint();
916.         stage.addChild(point);
917.
918.         point.x = transfCD(pArray[i], pMinW, pMaxW, pMin, pMax);
919.         point.y = transfCD(xArray[i], xMinW, xMaxW, xMin, xMax);
920.
921.
922.         point.visible = true;
923.         point.alpha = 1;
924.
925.         var join = new lib.line();
926.         stage.addChild(join);
927.
928.         if (i >= 1) {
929.             join.x = transfCD(pArray[i-1], pMinW, pMaxW, pMin, pMax);
930.             join.y = transfCD(xArray[i-1], xMinW, xMaxW, xMin, xMax);
931.             join.endX = transfCD(pArray[i], pMinW, pMaxW, pMin, pMax);
932.             join.endY = transfCD(xArray[i], xMinW, xMaxW, xMin, xMax);
933.
934.
935.             join.len = Math.sqrt(Math.pow((join.endY - join.y), 2) + Math.pow((join.endX -
join.x), 2));
936.
937.             join.scaleX = join.len;
938.
939.             join.rotation = Math.atan2((join.endY - join.y), (join.endX -
join.x)) * 180 / Math.PI;
940.
941.             join.visible = true;
942.             join.alpha = 1;
943.             join.addEventListener('tick', liveLine);
944.         }
945.
946.         point.visible = true;
947.         point.alpha = 1;
948.
949.         stage.addChild(point);
950.
951.
952.         point.addEventListener('tick', livePoint);
953.     }
954.
955.
956.     if (pMin < 0 && 1.2*pMax >= 0) {
957.         this.graphWindow.yAxis.x = transfCD(0, pMinW - 500, pMaxW - 500, pMin, pMax);

```

```

958.         }
959.         else {
960.             this.graphWindow.yAxis.x = -187.35;
961.         }
962.
963.
964.         if (xMin < 0 && 1.2*xMax >= 0) {
965.             this.graphWindow.xAxis.y = transfCD(0, xMinW - 370, xMaxW - 370, xMin, xMax);
966.         }
967.         else {
968.             this.graphWindow.xAxis.y = 203.25;
969.         }
970.
971.
972.
973.     }
974.
975. }
976.
977.
978. function transfCD(x, xStart, xEnd, xMin, xMax) {
979.     var newX = xStart + ((x-xMin) * (xEnd - xStart)/(xMax - xMin));
980.     return newX;
981. }
982.
983.
984. function livePoint(e) {
985.     var object = e.currentTarget;
986.
987.     if (!graphOpen) {
988.         object.alpha = 0;
989.         object.visible = false;
990.         object.removeEventListener('tick', livePoint);
991.         stage.removeChild(object);
992.     }
993.
994. }
995.
996.
997. function liveLine(e) {
998.     var object = e.currentTarget;
999.
1000.     if (!graphOpen) {
1001.         object.alpha = 0;
1002.         object.visible = false;
1003.         object.removeEventListener('tick', liveLine);
1004.         stage.removeChild(object);
1005.     }
1006.
1007. }
1008.
1009.
1010. function podstava(x, p, formule) {
1011.     if (formule.inner) {
1012.         for (var i = 0 ; i < formule.inner.length ; ++i) {
1013.             if (formule.inner[i].ist == "root") {
1014.                 formule.inner[i] = x;
1015.             }
1016.             else if (formule.inner[i].ist == "param") {
1017.                 formule.inner[i] = p;
1018.             }
1019.             else if (formule.inner[i].ist == "numb") {
1020.                 if (formule.inner[i].type == "e") {
1021.                     formule.inner[i] = 2.718281828459;
1022.                 }
1023.                 else if (formule.inner[i].type == "pi") {
1024.                     formule.inner[i] = 3.14159265359;
1025.                 }
1026.                 else {

```

```

1027.         formule.inner[i] = formule.inner[i].value;
1028.     }
1029. }
1030. else if (formule.inner[i].ist == "function") {
1031.     if (formule.inner[i].type == "log" || formule.inner[i].type == "root") {
1032.         formule.inner[i].arg = podstava(x, p, formule.inner[i].arg.inner);
1033.         formule.inner[i] = solveFun(formule.inner[i].type, podstava(x, p, formule.
1034.             inner[i].inner), formule.inner[i].arg);
1035.     }
1036.     else if (formule.inner[i].type == "power") {
1037.         if (formule.inner[i].arg.ist == "numb") {
1038.             if (formule.inner[i].arg.type == "e") {
1039.                 formule.inner[i].arg = 2.718281828459;
1040.             }
1041.             else if (formule.inner[i].arg.type == "pi") {
1042.                 formule.inner[i].arg = 3.14159265359;
1043.             }
1044.             else {
1045.                 formule.inner[i].arg = formule.inner[i].arg.value;
1046.             }
1047.         }
1048.         else if (formule.inner[i].arg.ist == "param") {
1049.             formule.inner[i].arg = p;
1050.         }
1051.         formule.inner[i] = solveFun(formule.inner[i].type, podstava(x, p, formule.
1052.             inner[i].inner), formule.inner[i].arg);
1053.     }
1054.     else {
1055.         formule.inner[i] = solveFun(formule.inner[i].type, podstava(x, p, formule.
1056.             inner[i].inner), 0);
1057.     }
1058. }
1059. }
1060.
1061. for (var i = 1 ; i < formule.inner.length ; ++i) {
1062.     if (formule.inner[i].ist == "power") {
1063.         formule.inner[i-1] = Math.pow(formule.inner[i-1], formule.inner[i].pow_to);
1064.         formule.inner.splice(i, 1);
1065.         --i;
1066.     }
1067. }
1068.
1069. for (var i = 1 ; i < formule.inner.length ; ++i) {
1070.     if (formule.inner[i].ist == "operator") {
1071.         if (formule.inner[i].type == "*") {
1072.             formule.inner[i-1] = formule.inner[i-1] * formule.inner[i+1];
1073.             formule.inner.splice(i, 2);
1074.             --i;
1075.         }
1076.         else if (formule.inner[i].type == "/") {
1077.             formule.inner[i-1] = formule.inner[i-1] / formule.inner[i+1];
1078.             formule.inner.splice(i, 2);
1079.             --i;
1080.         }
1081.     }
1082. }
1083.
1084. for (var i = 1 ; i < formule.inner.length ; ++i) {
1085.     if (formule.inner[i].ist == "operator") {
1086.         if (formule.inner[i].type == "+") {
1087.             formule.inner[i-1] = formule.inner[i-1] + formule.inner[i+1];
1088.             formule.inner.splice(i, 2);
1089.             --i;
1090.         }
1091.         else if (formule.inner[i].type == "-") {

```



```

1093.         formule.inner[i-1] = formule.inner[i-1] - formule.inner[i+1];
1094.         formule.inner.splice(i, 2);
1095.         --i;
1096.     }
1097. }
1098. }
1099.
1100.
1101.     return formule.inner[0];
1102.
1103. }
1104. else {
1105.     return formule;
1106. }
1107. }
1108.
1109.
1110.
1111.
1112. function solveFun(fType, num, arg) {
1113.     if (fType == "bracket") {
1114.         return num;
1115.     }
1116.     else if (fType == "sin") {
1117.         return Math.sin(num);
1118.     }
1119.     else if (fType == "cos") {
1120.         return Math.cos(num);
1121.     }
1122.     else if (fType == "tan") {
1123.         return Math.tan(num);
1124.     }
1125.     else if (fType == "asin") {
1126.         return Math.asin(num);
1127.     }
1128.     else if (fType == "acos") {
1129.         return Math.acos(num);
1130.     }
1131.     else if (fType == "atan") {
1132.         return Math.atan(num);
1133.     }
1134.     else if (fType == "atan") {
1135.         return Math.atan(num);
1136.     }
1137.     else if (fType == "power") {
1138.         return Math.pow(arg, num);
1139.     }
1140.     else if (fType == "log") {
1141.         return Math.log(num) / Math.log(arg);
1142.     }
1143.     else if (fType == "root") {
1144.         return Math.pow(num, (1/arg));
1145.     }
1146.     else if (fType == "ln") {
1147.         return Math.log(num);
1148.     }
1149.     else if (fType == "sqrt") {
1150.         return Math.pow(num, (1/2));
1151.     }
1152.     else if (fType == "cbrt") {
1153.         return Math.pow(num, (1/3));
1154.     }
1155. }
1156.
1157.
1158.
1159.
1160. this.addMinus.addEventListener("click", addMinus.bind(this));
1161. function addMinus() {

```

```

1162.         if (!hasMinus) {
1163.             hasMinus = true;
1164.         }
1165.         else hasMinus = false;
1166.     }
1167.
1168.
1169.     function bll(ss) {
1170.         if (ss) {
1171.             return -1;
1172.         }
1173.         else return 1;
1174.     }
1175.
1176.     this.add_n.addEventListener("click", add_nf.bind(this));
1177.     function add_nf() {
1178.         if (curPlace.inner.length == 0) {
1179.             if (isE) {
1180.                 curPlace.inner.push(new numb("e", 1));
1181.                 removeAllf();
1182.             }
1183.             else if (isPI) {
1184.                 curPlace.inner.push(new numb("pi", 1));
1185.                 removeAllf();
1186.             }
1187.             else {
1188.                 curPlace.inner.push(new numb("number", toTheNum(custN, custNafter, vedZeros, bll(h
asMinus))));
1189.                 removeAllf();
1190.             }
1191.         }
1192.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1193.             if (isE) {
1194.                 curPlace.inner.push(new numb("e", 1));
1195.                 removeAllf();
1196.             }
1197.             else if (isPI) {
1198.                 curPlace.inner.push(new numb("pi", 1));
1199.                 removeAllf();
1200.             }
1201.             else {
1202.                 if (curPlace.inner[curPlace.inner.length -
1].type != "/" || toTheNum(custN, custNafter, vedZeros, bll(hasMinus)) != 0) {
1203.                     curPlace.inner.push(new numb("number", toTheNum(custN, custNafter, vedZeros, b
ll(hasMinus))));
1204.                     removeAllf();
1205.                 }
1206.             }
1207.         }
1208.     }
1209.
1210.     this.add_p.addEventListener("click", add_pf.bind(this));
1211.     function add_pf() {
1212.         if (signsNow < 300) {
1213.             if (curPlace.inner.length == 0) {
1214.                 curPlace.inner.push(new par());
1215.             }
1216.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1217.                 curPlace.inner.push(new par());
1218.             }
1219.         }
1220.     }
1221.
1222.     this.add_x.addEventListener("click", add_xf.bind(this));
1223.     function add_xf() {
1224.         if (signsNow < 300) {
1225.             if (curPlace.inner.length == 0) {
1226.                 curPlace.inner.push(new root());
1227.             }

```

```

1228.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1229.             curPlace.inner.push(new root());
1230.         }
1231.     }
1232. }
1233.
1234. this.f_plus.addEventListener("click", f_plusf.bind(this));
1235. function f_plusf() {
1236.     if (signsNow < 300) {
1237.         if (curPlace.inner.length != 0) {
1238.             if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1239.                 curPlace.inner.push(new operator("+"));
1240.             }
1241.         }
1242.     }
1243. }
1244.
1245. this.f_minus.addEventListener("click", f_minusf.bind(this));
1246. function f_minusf() {
1247.     if (signsNow < 300) {
1248.         if (curPlace.inner.length != 0) {
1249.             if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1250.                 curPlace.inner.push(new operator("-"));
1251.             }
1252.         }
1253.     }
1254. }
1255.
1256. this.f_mult.addEventListener("click", f_multf.bind(this));
1257. function f_multf() {
1258.     if (signsNow < 300) {
1259.         if (curPlace.inner.length != 0) {
1260.             if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1261.                 curPlace.inner.push(new operator("*"));
1262.             }
1263.         }
1264.     }
1265. }
1266.
1267. this.f_div.addEventListener("click", f_divf.bind(this));
1268. function f_divf() {
1269.     if (signsNow < 300) {
1270.         if (curPlace.inner.length != 0) {
1271.             if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1272.                 curPlace.inner.push(new operator("/"));
1273.             }
1274.         }
1275.     }
1276. }
1277.
1278. this.powNop.addEventListener("click", powNopf.bind(this));
1279. function powNopf() {
1280.     if (signsNow < 300) {
1281.         if (curPlace.inner.length != 0) {
1282.             if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1283.                 if (isE) {
1284.                     curPlace.inner.push(new power([new numb("e", 1)], curPlace));
1285.                     removeAllf();
1286.                 }
1287.                 else if (isPI) {
1288.                     curPlace.inner.push(new power([new numb("pi", 1)], curPlace));
1289.                     removeAllf();
1290.                 }
1291.                 else {
1292.                     curPlace.inner.push(new power([new numb("number", toTheNum(custN, custNaft
1293. er, vedZeros, bll(hasMinus))], curPlace));
1294.                     removeAllf();
1295.                 }
1296.             }
1297.         }
1298.     }
1299. }

```

```

1296.     }
1297.   }
1298. }
1299.
1300. this.powPop.addEventListener("click", powPopf.bind(this));
1301. function powPopf() {
1302.   if (signsNow < 300) {
1303.     if (curPlace.inner.length != 0) {
1304.       if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1305.         curPlace.inner.push(new power([new par()], curPlace));
1306.       }
1307.     }
1308.   }
1309. }
1310.
1311. this.powPPNop.addEventListener("click", powPPNopf.bind(this));
1312. function powPPNopf() {
1313.   if (signsNow < 300) {
1314.     if (curPlace.inner.length != 0) {
1315.       if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1316.         if (isE) {
1317.           curPlace.inner.push(new power([new par(), new operator("+"), new numb("e",
1318. 1)], curPlace));
1319.           removeAllf();
1320.         } else if (isPI) {
1321.           curPlace.inner.push(new power([new par(), new operator("+"), new numb("pi"
1322. , 1)], curPlace));
1323.           removeAllf();
1324.         } else {
1325.           curPlace.inner.push(new power([new par(), new operator("+"), new numb("num
1326. ber", toTheNum(custN, custNafter, vedZeros, bl(hasMinus))], curPlace));
1327.           removeAllf();
1328.         }
1329.       }
1330.     }
1331.   }
1332. }
1333. this.powPMNop.addEventListener("click", powPMNopf.bind(this));
1334. function powPMNopf() {
1335.   if (signsNow < 300) {
1336.     if (curPlace.inner.length != 0) {
1337.       if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1338.         if (isE) {
1339.           curPlace.inner.push(new power([new par(), new operator("-
1340. "), new numb("e", 1)], curPlace));
1341.           removeAllf();
1342.         } else if (isPI) {
1343.           curPlace.inner.push(new power([new par(), new operator("-
1344. "), new numb("pi", 1)], curPlace));
1345.           removeAllf();
1346.         } else {
1347.           curPlace.inner.push(new power([new par(), new operator("-
1348. "), new numb("number", toTheNum(custN, custNafter, vedZeros, bl(hasMinus))], curPlace));
1349.           removeAllf();
1350.         }
1351.       }
1352.     }
1353.   }
1354. }
1355. this.powPxNop.addEventListener("click", powPxNopf.bind(this));
1356. function powPxNopf() {
1357.   if (signsNow < 300) {
1358.     if (curPlace.inner.length != 0) {

```

```

1359.         if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1360.             if (isE) {
1361.                 curPlace.inner.push(new power([new par(), new operator("*"), new numb("e",
1362.                     1)], curPlace));
1363.                 removeAllf();
1364.             }
1365.             else if (isPI) {
1366.                 curPlace.inner.push(new power([new par(), new operator("*"), new numb("pi"
1367.                     , 1)], curPlace));
1368.                 removeAllf();
1369.             }
1370.             else {
1371.                 curPlace.inner.push(new power([new par(), new operator("*"), new numb("num
1372.                     ber", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace));
1373.                 removeAllf();
1374.             }
1375.         }
1376.     }
1377.     this.powPdNop.addEventListener("click", powPdNopf.bind(this));
1378.     function powPdNopf() {
1379.         if (signsNow < 300) {
1380.             if (curPlace.inner.length != 0) {
1381.                 if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1382.                     if (isE) {
1383.                         curPlace.inner.push(new power([new par(), new operator("/"), new numb("e",
1384.                             1)], curPlace));
1385.                         removeAllf();
1386.                     }
1387.                     else if (isPI) {
1388.                         curPlace.inner.push(new power([new par(), new operator("/"), new numb("pi"
1389.                             , 1)], curPlace));
1390.                         removeAllf();
1391.                     }
1392.                     else {
1393.                         if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1394.                             curPlace.inner.push(new power([new par(), new operator("/"), new numb(
1395.                                 "number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace));
1396.                             removeAllf();
1397.                         }
1398.                     }
1399.                 }
1400.             }
1401.         }
1402.         this.nPowf.addEventListener("click", nPowff.bind(this));
1403.         function nPowff() {
1404.             if (signsNow < 300) {
1405.                 if (curPlace.inner.length == 0) {
1406.                     if (isE) {
1407.                         curPlace.inner.push(new funct("power", new numb("e", 1), curPlace));
1408.                         removeAllf();
1409.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1410.                     }
1411.                     else if (isPI) {
1412.                         curPlace.inner.push(new funct("power", new numb("pi", 1), curPlace));
1413.                         removeAllf();
1414.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1415.                     }
1416.                     else {
1417.                         if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1418.                             curPlace.inner.push(new funct("power", new numb("number", toTheNum(custN,
1419.                                 custNafter, vedZeros, bll(hasMinus))), curPlace));
1420.                             removeAllf();
1421.                             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1422.                         }
1423.                     }
1424.                 }
1425.             }
1426.         }
1427.     }

```

```

1421.     }
1422.   }
1423.   else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1424.     if (isE) {
1425.       curPlace.inner.push(new funct("power", new numb("e", 1), curPlace));
1426.       removeAllf();
1427.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1428.     }
1429.     else if (isPI) {
1430.       curPlace.inner.push(new funct("power", new numb("pi", 1), curPlace));
1431.       removeAllf();
1432.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1433.     }
1434.     else {
1435.       if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1436.         curPlace.inner.push(new funct("power", new numb("number", toTheNum(custN,
custNafter, vedZeros, bll(hasMinus))), curPlace));
1437.         removeAllf();
1438.         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1439.       }
1440.     }
1441.   }
1442. }
1443. }
1444.
1445.
1446. this.pPowf.addEventListener("click", pPowff.bind(this));
1447. function pPowff() {
1448.   if (signsNow < 300) {
1449.     if (curPlace.inner.length == 0) {
1450.       curPlace.inner.push(new funct("power", new par(), curPlace));
1451.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1452.     }
1453.     else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1454.       curPlace.inner.push(new funct("power", new par(), curPlace));
1455.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1456.     }
1457.   }
1458. }
1459.
1460.
1461. this.sinf.addEventListener("click", sinff.bind(this));
1462. function sinff() {
1463.   if (signsNow < 300) {
1464.     if (curPlace.inner.length == 0) {
1465.       curPlace.inner.push(new funct("sin", 0, curPlace));
1466.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1467.     }
1468.     else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1469.       curPlace.inner.push(new funct("sin", 0, curPlace));
1470.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1471.     }
1472.   }
1473. }
1474.
1475.
1476. this.asinf.addEventListener("click", asinff.bind(this));
1477. function asinff() {
1478.   if (signsNow < 300) {
1479.     if (curPlace.inner.length == 0) {
1480.       curPlace.inner.push(new funct("asin", 0, curPlace));
1481.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1482.     }
1483.     else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1484.       curPlace.inner.push(new funct("asin", 0, curPlace));
1485.       curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1486.     }
1487.   }
1488. }

```

```

1489.
1490.
1491. this.cosf.addEventListener("click", cosff.bind(this));
1492. function cosff() {
1493.     if (signsNow < 300) {
1494.         if (curPlace.inner.length == 0) {
1495.             curPlace.inner.push(new funct("cos", 0, curPlace));
1496.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1497.         }
1498.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1499.             curPlace.inner.push(new funct("cos", 0, curPlace));
1500.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1501.         }
1502.     }
1503. }
1504.
1505.
1506. this.acosf.addEventListener("click", acosff.bind(this));
1507. function acosff() {
1508.     if (signsNow < 300) {
1509.         if (curPlace.inner.length == 0) {
1510.             curPlace.inner.push(new funct("acos", 0, curPlace));
1511.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1512.         }
1513.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1514.             curPlace.inner.push(new funct("acos", 0, curPlace));
1515.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1516.         }
1517.     }
1518. }
1519.
1520.
1521. this.tanf.addEventListener("click", tanff.bind(this));
1522. function tanff() {
1523.     if (signsNow < 300) {
1524.         if (curPlace.inner.length == 0) {
1525.             curPlace.inner.push(new funct("tan", 0, curPlace));
1526.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1527.         }
1528.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1529.             curPlace.inner.push(new funct("tan", 0, curPlace));
1530.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1531.         }
1532.     }
1533. }
1534.
1535.
1536. this.atanf.addEventListener("click", atanff.bind(this));
1537. function atanff() {
1538.     if (signsNow < 300) {
1539.         if (curPlace.inner.length == 0) {
1540.             curPlace.inner.push(new funct("atan", 0, curPlace));
1541.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1542.         }
1543.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1544.             curPlace.inner.push(new funct("atan", 0, curPlace));
1545.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1546.         }
1547.     }
1548. }
1549.
1550.
1551.
1552. this.ex_form.addEventListener("click", ex_formf.bind(this));
1553. function ex_formf() {
1554.     if (signsNow < 300) {
1555.         if (curPlace.outer != null && curPlace.inner.length > 0) {
1556.             if (curPlace.inner[curPlace.inner.length - 1].ist != "operator") {
1557.                 curPlace.outer.inner[curPlace.outer.inner.length - 1].exit = true;

```

```

1558.         curPlace = curPlace.outer;
1559.     }
1560. }
1561. }
1562. }
1563.
1564.
1565. this.brac.addEventListener("click", bracf.bind(this));
1566. function bracf() {
1567.     if (signsNow < 300) {
1568.         if (curPlace.inner.length == 0) {
1569.             curPlace.inner.push(makeBracket([], curPlace));
1570.             curPlace.inner[curPlace.inner.length - 1].exit = false;
1571.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1572.         }
1573.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1574.             curPlace.inner.push(makeBracket([], curPlace));
1575.             curPlace.inner[curPlace.inner.length - 1].exit = false;
1576.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1577.         }
1578.     }
1579. }
1580.
1581.
1582. this.epowf.addEventListener("click", epowff.bind(this));
1583. function epowff() {
1584.     if (signsNow < 300) {
1585.         if (curPlace.inner.length == 0) {
1586.             curPlace.inner.push(new funct("power", new numb("e", 1), curPlace));
1587.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1588.         }
1589.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1590.             curPlace.inner.push(new funct("power", new numb("e", 1), curPlace));
1591.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1592.         }
1593.     }
1594. }
1595.
1596.
1597. this.logPPNf.addEventListener("click", logPPNff.bind(this));
1598. function logPPNff() {
1599.     if (signsNow < 300) {
1600.         if (curPlace.inner.length == 0) {
1601.             if (isE) {
1602.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("+")
1603. , new numb("e", 1)], curPlace), curPlace));
1604.                 removeAllf();
1605.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1606.             }
1607.             else if (isPI) {
1608.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("+")
1609. , new numb("pi", 1)], curPlace), curPlace));
1610.                 removeAllf();
1611.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1612.             }
1613.             else {
1614.                 if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1615.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new operator(
1616. "+"), new numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1617.                     removeAllf();
1618.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1619.                 }
1620.             }
1621.         }
1622.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1623.             if (isE) {
1624.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("+")
1625. , new numb("e", 1)], curPlace), curPlace));
1626.                 removeAllf();

```



```

1623.         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1624.     }
1625.     else if (isPI) {
1626.         curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("+")
1627. , new numb("pi", 1)], curPlace), curPlace));
1628.         removeAllf();
1629.         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1630.     }
1631.     else {
1632.         if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1633.             curPlace.inner.push(new funct("log", makeBracket([new par(), new operator(
1634. "+"), new numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1635.             removeAllf();
1636.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1637.         }
1638.     }
1639. }
1640.
1641.
1642. this.logPxNf.addEventListener("click", logPxNff.bind(this));
1643. function logPxNff() {
1644.     if (signsNow < 300) {
1645.         if (curPlace.inner.length == 0) {
1646.             if (isE) {
1647.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("*")
1648. , new numb("e", 1)], curPlace), curPlace));
1649.                 removeAllf();
1650.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1651.             }
1652.             else if (isPI) {
1653.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("*")
1654. , new numb("pi", 1)], curPlace), curPlace));
1655.                 removeAllf();
1656.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1657.             }
1658.             else {
1659.                 if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1660.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new operator(
1661. "*"), new numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1662.                     removeAllf();
1663.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1664.                 }
1665.             }
1666.         }
1667.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1668.             if (isE) {
1669.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("*")
1670. , new numb("e", 1)], curPlace), curPlace));
1671.                 removeAllf();
1672.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1673.             }
1674.             else if (isPI) {
1675.                 curPlace.inner.push(new funct("log", makeBracket([new par(), new operator("*")
1676. , new numb("pi", 1)], curPlace), curPlace));
1677.                 removeAllf();
1678.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1679.             }
1680.             else {
1681.                 if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1682.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new operator(
1683. "*"), new numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1684.                     removeAllf();
1685.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1686.                 }
1687.             }
1688.         }
1689.     }
1690. }

```

```

1684.     }
1685.
1686.
1687.     this.lnff.addEventListener("click", lnff.bind(this));
1688.     function lnff() {
1689.         if (signsNow < 300) {
1690.             if (curPlace.inner.length == 0) {
1691.                 curPlace.inner.push(new funct("ln", 1, curPlace));
1692.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1693.             }
1694.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1695.                 curPlace.inner.push(new funct("ln", 1, curPlace));
1696.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1697.             }
1698.         }
1699.     }
1700.
1701.
1702.     this.lognff.addEventListener("click", lognff.bind(this));
1703.     function lognff() {
1704.         if (signsNow < 300) {
1705.             if (curPlace.inner.length == 0) {
1706.                 if (isE) {
1707.                     curPlace.inner.push(new funct("ln", 1, curPlace));
1708.                     removeAllf();
1709.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1710.                 }
1711.                 else if (isPI) {
1712.                     curPlace.inner.push(new funct("log", makeBracket([new numb("pi", 1)], curPlace
1713. ), curPlace));
1714.                     removeAllf();
1715.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1716.                 }
1717.                 else {
1718.                     if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) > 0 && (toTheNum(cu
1719. stN, custNafter, vedZeros, bll(hasMinus))) != 1) {
1720.                         curPlace.inner.push(new funct("log", makeBracket([new numb("number", toThe
1721. Num(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1722.                         removeAllf();
1723.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1724.                     }
1725.                 }
1726.             }
1727.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1728.                 if (isE) {
1729.                     curPlace.inner.push(new funct("ln", 1, curPlace));
1730.                     removeAllf();
1731.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1732.                 }
1733.                 else if (isPI) {
1734.                     curPlace.inner.push(new funct("log", makeBracket([new numb("pi", 1)], curPlace
1735. ), curPlace));
1736.                     removeAllf();
1737.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1738.                 }
1739.                 else {
1740.                     if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1741.                         curPlace.inner.push(new funct("log", makeBracket([new numb("number", toThe
1742. Num(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1743.                         removeAllf();
1744.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1745.                     }
1746.                 }
1747.             }
1748.         }
1749.     }
1750.
1751.     this.logpff.addEventListener("click", logpff.bind(this));

```

```

1748.     function logpff() {
1749.         if (signsNow < 300) {
1750.             if (curPlace.inner.length == 0) {
1751.                 curPlace.inner.push(new funct("log", makeBracket([new par()], curPlace), curPlace)
1752.             );
1753.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1754.             }
1755.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1756.                 curPlace.inner.push(new funct("log", makeBracket([new par()], curPlace), curPlace)
1757.             );
1758.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1759.             }
1760.         }
1761.     }
1762.     this.loppownf.addEventListener("click", loppownff.bind(this));
1763.     function loppownff() {
1764.         if (signsNow < 300) {
1765.             if (curPlace.inner.length == 0) {
1766.                 if (isE) {
1767.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new power([new nu
1768. mb("e", 1)], curPlace)], curPlace), curPlace));
1769.                     removeAllf();
1770.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1771.                 }
1772.                 else if (isPI) {
1773.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new power([new nu
1774. mb("pi", 1)], curPlace)], curPlace), curPlace));
1775.                     removeAllf();
1776.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1777.                 }
1778.                 else {
1779.                     if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1780.                         curPlace.inner.push(new funct("log", makeBracket([new par(), new power([ne
1781. w numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace)], curPlace), curPlace
1782. ));
1783.                         removeAllf();
1784.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1785.                     }
1786.                 }
1787.             }
1788.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1789.                 if (isE) {
1790.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new power([new nu
1791. mb("e", 1)], curPlace)], curPlace), curPlace));
1792.                     removeAllf();
1793.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1794.                 }
1795.                 else if (isPI) {
1796.                     curPlace.inner.push(new funct("log", makeBracket([new par(), new power([new nu
1797. mb("pi", 1)], curPlace)], curPlace), curPlace));
1798.                     removeAllf();
1799.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1800.                 }
1801.                 else {
1802.                     if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1803.                         curPlace.inner.push(new funct("log", makeBracket([new par(), new power([ne
1804. w numb("number", toTheNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace)], curPlace), curPlace
1805. ));
1806.                         removeAllf();
1807.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1808.                     }
1809.                 }
1810.             }
1811.         }
1812.     }
1813. }

```

```

1807.
1808.
1809.     this.sqrtof.addEventListener("click", sqrtoff.bind(this));
1810.     function sqrtoff() {
1811.         if (signsNow < 300) {
1812.             if (curPlace.inner.length == 0) {
1813.                 curPlace.inner.push(new funct("sqrt", 1, curPlace));
1814.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1815.             }
1816.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1817.                 curPlace.inner.push(new funct("sqrt", 1, curPlace));
1818.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1819.             }
1820.         }
1821.     }
1822.
1823.
1824.
1825.
1826.     this.cbrtof.addEventListener("click", cbrtoff.bind(this));
1827.     function cbrtoff() {
1828.         if (signsNow < 300) {
1829.             if (curPlace.inner.length == 0) {
1830.                 curPlace.inner.push(new funct("cbrt", 1, curPlace));
1831.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1832.             }
1833.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1834.                 curPlace.inner.push(new funct("cbrt", 1, curPlace));
1835.                 curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1836.             }
1837.         }
1838.     }
1839.
1840.
1841.     this.nrootof.addEventListener("click", nrootof.bind(this));
1842.     function nrootof() {
1843.         if (signsNow < 300) {
1844.             if (curPlace.inner.length == 0) {
1845.                 if (isE) {
1846.                     curPlace.inner.push(new funct("root", makeBracket([new numb("e", 1)], curPlace
1847. ), curPlace));
1848.                     removeAllf();
1849.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1850.                 }
1851.                 else if (isPI) {
1852.                     curPlace.inner.push(new funct("root", makeBracket([new numb("pi", 1)], curPlac
1853. e), curPlace));
1854.                     removeAllf();
1855.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1856.                 }
1857.                 else {
1858.                     if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1859.                         curPlace.inner.push(new funct("root", makeBracket([new numb("number", toTh
1860. eNum(custN, custNafter, vedZeros, bll(hasMinus))], curPlace), curPlace));
1861.                         removeAllf();
1862.                         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1863.                     }
1864.                 }
1865.             }
1866.             else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1867.                 if (isE) {
1868.                     curPlace.inner.push(new funct("root", makeBracket([new numb("e", 1)], curPlace
1869. ), curPlace));
1870.                     removeAllf();
1871.                     curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1872.                 }
1873.                 else if (isPI) {
1874.                     curPlace.inner.push(new funct("root", makeBracket([new numb("pi", 1)], curPlac
1875. e), curPlace));
1876.                 }
1877.             }
1878.         }
1879.     }
1880.
1881.
1882.
1883.
1884.
1885.
1886.
1887.
1888.
1889.
1890.
1891.
1892.
1893.
1894.
1895.
1896.
1897.
1898.
1899.
1900.

```

```

1871.         removeAllf();
1872.         curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1873.     }
1874.     else {
1875.         if ((toTheNum(custN, custNafter, vedZeros, bll(hasMinus))) != 0) {
1876.             curPlace.inner.push(new funct("root", makeBracket([new numb("number", toTh
eNum(custN, custNafter, vedZeros,bll(hasMinus))]), curPlace), curPlace));
1877.             removeAllf();
1878.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1879.         }
1880.     }
1881. }
1882. }
1883. }
1884.
1885.
1886. this.prootff.addEventListener("click", prootff.bind(this));
1887. function prootff() {
1888.     if (signsNow < 300) {
1889.         if (curPlace.inner.length == 0) {
1890.             curPlace.inner.push(new funct("root", makeBracket([new par()], curPlace), curPlace
));
1891.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1892.         }
1893.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
1894.             curPlace.inner.push(new funct("root", makeBracket([new par()], curPlace), curPlace
));
1895.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
1896.         }
1897.     }
1898. }
1899.
1900.
1901. this.makeDerf.addEventListener("click", makeDerf.bind(this));
1902. function makeDerf() {
1903.     if (signsNow < 300) {
1904.         for (var i = 0 ; i < 10 ; ++i) {
1905.             formula.inner = optimize(formula.inner, 0);
1906.         }
1907.         formula = findDerivative(formula, formula.outer);
1908.         for (var i = 0 ; i < 10 ; ++i) {
1909.             formula.inner = optimize(formula.inner, 0);
1910.         }
1911.         curPlace = formula;
1912.     }
1913. }
1914.
1915.
1916.
1917.
1918. function findDerivative(formula, theOuter) {
1919.     var index = 0;
1920.     var total = new theFormula([], theOuter);
1921.     if (formula.inner.length == 2 && formula.inner[1].ist == "power") {
1922.         return getDer(formula, theOuter);
1923.     }
1924.     else if (formula.inner.length > 0) {
1925.         while (index < formula.inner.length) {
1926.             var partArray = new theFormula([], theOuter);
1927.             var reversed = false;
1928.             while (index < formula.inner.length && (formula.inner[index].ist != "operator" ||
(formula.inner[index].type != "+" && formula.inner[index].type != "-"))) {
1929.                 if (formula.inner[index].ist == "operator" && formula.inner[index].type == "*"
) {
1930.                     partArray.inner.push(formula.inner[index]);
1931.                 }
1932.                 else if (formula.inner[index].ist == "operator" && formula.inner[index].type =
= "/" ) {
1933.                     partArray.inner.push(new operator("*"));

```

```

1934.         reversed = true;
1935.     }
1936.     else if (formula.inner[index].ist == "power") {
1937.         partArray.inner.push(formula.inner[index]);
1938.     }
1939.     else if (reversed) {
1940.         if (formula.inner[index].ist == "root") {
1941.             partArray.inner.push(formula.inner[index]);
1942.             partArray.inner.push(new power([new numb("number", -1)], partArray));
1943.             reversed = false;
1944.         }
1945.         else if (formula.inner[index].ist == "param") {
1946.             partArray.inner.push(formula.inner[index]);
1947.             partArray.inner.push(new power([new numb("number", -1)], partArray));
1948.             reversed = false;
1949.         }
1950.         else if (formula.inner[index].ist == "numb") {
1951.             if (formula.inner[index].type == "e") {
1952.                 partArray.inner.push(new numb("number", (1/2.718281828459)));
1953.                 partArray.inner.push(new power([new numb("number", 1)], partArray)
1954.             );
1955.                 reversed = false;
1956.             }
1957.             else if (formula.inner[index].type == "pi") {
1958.                 partArray.inner.push(new numb("number", (1/3.141592653589)));
1959.                 partArray.inner.push(new power([new numb("number", 1)], partArray)
1960.             );
1961.                 reversed = false;
1962.             }
1963.             else if (formula.inner[index].type == "number") {
1964.                 partArray.inner.push(new numb("number", (1/formula.inner[index].va
1965. lue)));
1966.                 partArray.inner.push(new power([new numb("number", 1)], partArray)
1967.             );
1968.                 reversed = false;
1969.             }
1970.         }
1971.         else if (formula.inner[index].ist == "function") {
1972.             partArray.inner.push(formula.inner[index]);
1973.             partArray.inner.push(new power([new numb("number", -1)], partArray));
1974.             reversed = false;
1975.         }
1976.     }
1977.     else {
1978.         partArray.inner.push(formula.inner[index]);
1979.         partArray.inner.push(new power([new numb("number", 1)], partArray));
1980.     }
1981.     ++index;
1982. }
1983.
1984. //shrink double powers
1985. for (var i = 1 ; i < partArray.inner.length ; ++i) {
1986.     if (partArray.inner[i-
1987. 1].ist == "power" && partArray.inner[i].ist == "power") {
1988.         var newPower = new power([], partArray);
1989.
1990.         var part1 = partArray.inner[i-1].pow_to.inner;
1991.         var brPart1 = makeBracket(part1, partArray);
1992.
1993.         newPower.pow_to.inner.push(brPart1);
1994.         newPower.pow_to.inner.push(new operator("*"));
1995.
1996.         var part2 = partArray.inner[i].pow_to.inner;
1997.         var brPart2 = makeBracket(part2, partArray);
1998.
1999.         newPower.pow_to.inner.push(brPart2);
2000.         partArray.inner[i-1] = newPower;
2001.         partArray.inner.splice(i, 1);
2002.         --i;

```

```

1998.     }
1999.     }
2000.
2001.     var derivResult = new theFormula([], total);
2002.     var firstVal = new theFormula([], derivResult);
2003.     var firstDer = new theFormula([], derivResult);
2004.     var secondVal = new theFormula([], derivResult);
2005.     var secondDer = new theFormula([], derivResult);
2006.
2007.     firstVal.inner.push(partArray.inner[0]);
2008.
2009.     for (var i = 0 ; i < 5 ; ++i) {
2010.         optimize(partArray.inner[1], 0);
2011.     }
2012.     firstVal.inner.push(partArray.inner[1]);
2013.
2014.     partArray.inner.splice(0, 3);
2015.
2016.     secondVal = partArray;
2017.
2018.     if (partArray.inner.length == 2) {
2019.         for (var i = 0 ; i < 5 ; ++i) {
2020.             optimize(partArray.inner[1].pow_to.inner, 0);
2021.         }
2022.     }
2023.
2024.
2025.     firstDer = getDer(firstVal, derivResult);
2026.     secondDer = getDer(secondVal, derivResult);
2027.
2028.
2029.     for (var i = 0 ; i < 5 ; ++i) {
2030.         optimize(firstDer.inner, 0);
2031.         optimize(secondDer.inner, 0);
2032.     }
2033.
2034.
2035.     if (secondVal.inner.length > 0) {
2036.         derivResult.inner.push(makeBracket(firstDer.inner, derivResult));
2037.         derivResult.inner.push(new operator("*"));
2038.         derivResult.inner.push(makeBracket(secondVal.inner, derivResult));
2039.         derivResult.inner.push(new operator("+"));
2040.         derivResult.inner.push(makeBracket(secondDer.inner, derivResult));
2041.         derivResult.inner.push(new operator("*"));
2042.         derivResult.inner.push(makeBracket(firstVal.inner, derivResult));
2043.     }
2044.     else {
2045.         derivResult.inner.push(makeBracket(firstDer.inner, derivResult));
2046.     }
2047.
2048.
2049.     for (var i = 0 ; i < 10 ; ++i) {
2050.         //optimize(derivResult.inner, 0);
2051.     }
2052.
2053.
2054.     total.inner.push(makeBracket(derivResult.inner, total));
2055.
2056.     if (index < formula.inner.length) {
2057.         if (formula.inner[index].ist == "operator" && formula.inner[index].type == "+"
2058.     ) {
2059.             total.inner.push(new operator("+"));
2060.         }
2061.         else if (formula.inner[index].ist == "operator" && formula.inner[index].type =
2062.     = "-") {
2063.             total.inner.push(new operator("-"));
2064.         }
2065.     }

```

```

2065.         ++index;
2066.     }
2067. }
2068.
2069.     return total;
2070.
2071. }
2072. }
2073.
2074.
2075.
2076.
2077.
2078. function getDer(formula, out) {
2079.     if (formula.inner.length == 2) {
2080.         var result = new theFormula([], out);
2081.
2082.         var powFormula = copyF(formula.inner[1].pow_to);
2083.
2084.         var inBracket1 = makeBracket(copyF(powFormula).inner, result);
2085.
2086.
2087.         for (var i = 0; i < 10 ; ++i) {
2088.             inBracket1.inner.inner = optimize(inBracket1.inner.inner, 0);
2089.         }
2090.
2091.
2092.         result.inner.push(inBracket1);
2093.         result.inner.push(new operator("*"));
2094.
2095.
2096.         result.inner.push(copyThis(formula.inner[0], result));
2097.
2098.
2099.
2100.         var inBracket2 = makeBracket(copyF(powFormula).inner, result);
2101.
2102.
2103.         var newOpted = [inBracket2, new operator("-"), new numb("number", 1)]
2104.
2105.         for (var i = 0; i < 10 ; ++i) {
2106.             newOpted = optimize(newOpted, 0);
2107.         }
2108.         result.inner.push(new power(newOpted, result));
2109.
2110.         if (formula.inner[0].ist == "function") {
2111.             result.inner.push(new operator("*"));
2112.             var brDerInner = makeBracket(findDerivative(formula.inner[0].inner, result).inner,
result);
2113.             result.inner.push(brDerInner);
2114.         }
2115.         result.inner.push(new operator("*"));
2116.
2117.         result.inner.push(funcDer(formula.inner[0], result));
2118.
2119.         return result;
2120.     }
2121.     else if (formula.inner.length < 2) {
2122.         return new theFormula([new numb("number", 0)], null);
2123.     }
2124.     else {
2125.         return findDerivative(formula, formula.outer);
2126.     }
2127. }
2128.
2129.
2130.
2131.
2132.

```



```

2133.     function funcDer(object, nOuter) {
2134.         if (object.ist == "root") {
2135.             return new numb("number", 1);
2136.         }
2137.         else if (object.ist == "param") {
2138.             return new numb("number", 0);
2139.         }
2140.         else if (object.ist == "numb") {
2141.             return new numb("number", 0);
2142.         } //производные функций
2143.         else if (object.ist == "function") {
2144.             if (object.type == "bracket") {
2145.                 return new numb("number", 1);
2146.             }
2147.             else if (object.type == "sin") {
2148.                 var newBRF = makeBracket([], nOuter);
2149.                 newBRF.exit = true;
2150.                 var newFF = new funct("cos", 0, newBRF.inner);
2151.                 newFF.inner.inner = copyArr(object.inner.inner, newFF.inner);
2152.                 newFF.exit = true;
2153.                 newBRF.inner = new theFormula([new numb("number", 1), new operator("*"), newFF], n
Outer);
2154.                 newFF.inner.outer = newBRF.inner;
2155.
2156.                 return newBRF;
2157.             }
2158.             else if (object.type == "cos") {
2159.                 var newBRF = makeBracket([], nOuter);
2160.                 newBRF.exit = true;
2161.                 var newFF = new funct("sin", 0, newBRF.inner);
2162.                 newFF.inner.inner = copyArr(object.inner.inner, newFF.inner);
2163.                 newFF.exit = true;
2164.                 newBRF.inner = new theFormula([new numb("number", -
1), new operator("*"), newFF], nOuter);
2165.                 newFF.inner.outer = newBRF.inner;
2166.
2167.                 return newBRF;
2168.             }
2169.             else if (object.type == "tan") {
2170.                 var newBRF = makeBracket([], nOuter);
2171.                 newBRF.exit = true;
2172.                 var newFF = new funct("cos", 0, newBRF.inner);
2173.                 newFF.inner.inner = copyArr(object.inner.inner, newFF.inner);
2174.                 newFF.exit = true;
2175.                 newBRF.inner = new theFormula([new numb("number", 1), new operator("/"), makeBrack
et([newFF, new power([new numb("number", 2], newBRF.inner)]), nOuter);
2176.                 newFF.inner.outer = newBRF.inner;
2177.
2178.                 return newBRF;
2179.             }
2180.             else if (object.type == "asin") {
2181.                 var newBRF = makeBracket([], nOuter);
2182.                 newBRF.exit = true;
2183.                 var newInner2 = new theFormula([], nOuter)
2184.                 var newInner = new theFormula([], newInner2)
2185.                 newInner.inner = [new numb("number", 1), new operator("-
"), makeBracket(copyArr(object.inner.inner, newInner), newInner), newpower([new numb("number", 2)], 1)]
;
2186.                 var newFF = new funct("sqrt", 0, newInner.inner);
2187.                 newFF.exit = true;
2188.
2189.                 newFF.inner.inner = copyArr(newInner.inner, newFF.inner);
2190.                 newInner2.inner = [new numb("number", 1), new operator("/"), newFF];
2191.                 newBRF.inner = copyF(newInner2);
2192.
2193.                 return newBRF;
2194.             }
2195.             else if (object.type == "acos") {
2196.                 var newBRF = makeBracket([], nOuter);

```

```

2197.         newBRF.exit = true;
2198.         var newInner2 = new theFormula([], nOuter)
2199.         var newInner = new theFormula([], newInner2)
2200.         newInner.inner = [new numb("number", 1), new operator("-
    "), makeBracket(copyArr(object.inner.inner, newInner), newInner), newpower([new numb("number", 2)], 1)]
    ];
2201.         var newFF = new funct("sqrt", 0, newInner.inner);
2202.         newFF.exit = true;
2203.
2204.         newFF.inner.inner = copyArr(newInner.inner, newFF.inner);
2205.         newInner2.inner = [new numb("number", -1), new operator("/"), newFF];
2206.         newBRF.inner = copyF(newInner2);
2207.
2208.         return newBRF;
2209.     }
2210.     else if (object.type == "atan") {
2211.         var newBRF = makeBracket([], nOuter);
2212.         newBRF.exit = true;
2213.         var newInner2 = new theFormula([], nOuter)
2214.         var newInner = new theFormula([], newInner2)
2215.         newInner.inner = [new numb("number", 1), new operator("+"), makeBracket(copyArr(object.inner.inner, newInner), newInner), newpower([new numb("number", 2)], 1)];
2216.         var newFF = makeBracket([], newInner.inner);
2217.         newFF.inner.inner = copyArr(newInner.inner, newFF.inner);
2218.         newInner2.inner = [new numb("number", 1), new operator("/"), newFF];
2219.         newBRF.inner = copyF(newInner2);
2220.         return newBRF;
2221.     }
2222.     else if (object.type == "power") {
2223.         var newBRF = makeBracket([], nOuter);
2224.         newBRF.exit = true;
2225.
2226.         var objCopy = copyThis(object, newBRF.inner);
2227.
2228.         var lnFunc = new funct("ln", 1, newBRF.inner);
2229.         lnFunc.inner.inner = [object.arg];
2230.         lnFunc.exit = true;
2231.
2232.         newBRF.inner.inner = [copyThis(objCopy, newBRF.inner), new operator("*"), lnFunc];
2233.
2234.
2235.         return newBRF;
2236.     }
2237.     else if (object.type == "ln") {
2238.         var newBRF = makeBracket([], nOuter);
2239.         newBRF.exit = true;
2240.         var newInner = new theFormula([], nOuter);
2241.
2242.         var inF = copyF(object.inner);
2243.         var newBr = makeBracket(inF.inner, newBRF.inner);
2244.
2245.         newInner.inner = [new numb("number", 1), new operator("/"), newBr];
2246.
2247.         newBRF.inner = copyF(newInner);
2248.
2249.         return newBRF;
2250.     }
2251.     else if (object.type == "log") {
2252.         var newBRF = makeBracket([], nOuter);
2253.         newBRF.exit = true;
2254.         var newInner = new theFormula([], nOuter);
2255.
2256.         var inF = copyF(object.inner);
2257.         var newBr = makeBracket(copyArr(inF.inner), newBRF.inner);
2258.
2259.         var brackUnder = makeBracket([], newBRF.inner);
2260.
2261.         var dopLn = new funct("ln", 0, brackUnder.inner);
2262.

```

```

2263.         dopLn.inner = copyF(object.arg.inner);
2264.
2265.         brackUnder.inner.inner = [newBr, new operator("*"), dopLn];
2266.
2267.         newInner.inner = [new numb("number", 1), new operator("/"), brackUnder];
2268.
2269.         newBRF.inner = copyF(newInner);
2270.
2271.         return newBRF;
2272.     }
2273.     else if (object.type == "sqrt") {
2274.         var newBRF = makeBracket([], nOuter);
2275.         newBRF.exit = true;
2276.
2277.         var objCopy = copyThis(object, newBRF.inner);
2278.
2279.         var brack = makeBracket([], newBRF.inner);
2280.
2281.         brack.inner.inner = [new numb("number", 2), new operator("*"), copyThis(objCopy, brack.inner)];
2282.
2283.         newBRF.inner.inner = [new numb("number", 1), new operator("/"), copyThis(brack, newBRF.inner)];
2284.
2285.
2286.         return newBRF;
2287.     }
2288.     else if (object.type == "cbrt") {
2289.         var newBRF = makeBracket([], nOuter);
2290.         newBRF.exit = true;
2291.
2292.
2293.         var objCopy = copyThis(object, newBRF.inner);
2294.         var powBracket = makeBracket([], objCopy.inner);
2295.
2296.         powBracket.inner = copyF(objCopy.inner);
2297.         objCopy.inner.inner = [copyThis(powBracket), new power([new numb("number", 2)], objCopy.inner)];
2298.
2299.         var brack = makeBracket([], newBRF.inner);
2300.
2301.         brack.inner.inner = [new numb("number", 3), new operator("*"), copyThis(objCopy, brack.inner)];
2302.
2303.         newBRF.inner.inner = [new numb("number", 1), new operator("/"), copyThis(brack, newBRF.inner)];
2304.
2305.
2306.         return newBRF;
2307.     }
2308.     else if (object.type == "root") {
2309.         var newBRF = makeBracket([], nOuter);
2310.         newBRF.exit = true;
2311.
2312.
2313.         var objCopy = copyThis(object, newBRF.inner);
2314.         var powBracket = makeBracket([], objCopy.inner);
2315.
2316.         powBracket.inner = copyF(objCopy.inner);
2317.         objCopy.inner.inner = [copyThis(powBracket), new power([copyThis(object.arg.inner.inner[0]), new operator("-"), new numb("number", 1)], objCopy.inner)];
2318.
2319.         var brack = makeBracket([], newBRF.inner);
2320.
2321.         brack.inner.inner = [copyThis(object.arg.inner.inner[0]), new operator("*"), copyThis(objCopy, brack.inner)];
2322.
2323.         newBRF.inner.inner = [new numb("number", 1), new operator("/"), copyThis(brack, newBRF.inner)];

```

```

2324.
2325.
2326.         return newBRF;
2327.     }
2328. }
2329. }
2330.
2331.
2332.
2333.
2334.
2335. this.deleteOne.addEventListener("click", deleteOnef.bind(this));
2336. function deleteOnef() {
2337.     if (curPlace.inner.length > 0) {
2338.         if (curPlace.inner[curPlace.inner.length - 1].ist == "operator") {
2339.             curPlace.inner.pop();
2340.         }
2341.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "root") {
2342.             curPlace.inner.pop();
2343.         }
2344.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "param") {
2345.             curPlace.inner.pop();
2346.         }
2347.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "numb") {
2348.             curPlace.inner.pop();
2349.         }
2350.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "power") {
2351.             curPlace.inner.pop();
2352.         }
2353.         else if (curPlace.inner[curPlace.inner.length - 1].ist == "function") {
2354.             curPlace.inner[curPlace.inner.length - 1].exit = false;
2355.             curPlace = curPlace.inner[curPlace.inner.length - 1].inner;
2356.         }
2357.     }
2358.     else if (curPlace.inner.length == 0 && curPlace.outer != null) {
2359.         curPlace = curPlace.outer;
2360.         curPlace.inner.pop();
2361.     }
2362.
2363.     //Я без понятия, как фиксить эту х*йню с выходами из функций
2364.     //Уже и так несколько часов сижу, но иногда норм не удаляются
2365.     //Легче нажать clear и ввести формулу заново
2366.     //А вообще лучше не преобразовывать функцию в производную
2367.     //От этого одни проблемы
2368.
2369.     //Ибо эта проблема встречается после преобразования в производную
2370.     //Если сам вводил формулу, то все норм удаляется
2371.
2372. }
2373.
2374. this.clearAll.addEventListener("click", clearAllf.bind(this));
2375. function clearAllf() {
2376.     formula = new theFormula([], null);
2377.     curPlace = formula;
2378. }
2379.
2380.
2381.
2382. this.optimizeB.addEventListener("click", optimizeBf.bind(this));
2383. function optimizeBf() {
2384.     formula.inner = optimize(formula.inner, 0);
2385. }
2386.
2387.
2388.
2389. function optimize(thisArr, startPos) {
2390.     if (thisArr.length >= 3) {
2391.         for (var i = 2 + startPos ; i < thisArr.length ; ++i) {
2392.             var prev2 = thisArr[i-2];

```

```

2393.     var prev = thisArr[i-1];
2394.     var thisX = thisArr[i];
2395.
2396.     if (prev.ist == "operator") {
2397.         if (prev.type == "*") {
2398.             if (thisArr.length <= i+1) {
2399.                 if (prev2.ist == "numb" && thisX.ist == "numb") {
2400.                     if (prev2.type == "number" && thisX.type == "number") {
2401.                         prev2.value = Math.round(prev2.value * thisX.value*1000000000)
2402.                         /1000000000;
2403.                         thisArr.splice(i-1, 2);
2404.                         i -= 1;
2405.                     }
2406.                 }
2407.             }
2408.         } else if (thisArr[i+1].ist != "power") {
2409.             if (prev2.ist == "numb" && thisX.ist == "numb") {
2410.                 if (prev2.type == "number" && thisX.type == "number") {
2411.                     prev2.value = Math.round(prev2.value * thisX.value*1000000000)
2412.                     /1000000000;
2413.                     thisArr.splice(i-1, 2);
2414.                     i -= 1;
2415.                 }
2416.             }
2417.         } else if (prev.type == "/") {
2418.             if (thisArr.length <= i+1) {
2419.                 if (prev2.ist == "numb" && thisX.ist == "numb") {
2420.                     if (prev2.type == "number" && thisX.type == "number") {
2421.                         prev2.value = Math.round(prev2.value / thisX.value*1000000000)
2422.                         /1000000000;
2423.                         thisArr.splice(i-1, 2);
2424.                         i -= 1;
2425.                     }
2426.                 }
2427.             }
2428.         } else if (thisArr[i+1].ist != "power") {
2429.             if (prev2.ist == "numb" && thisX.ist == "numb") {
2430.                 if (prev2.type == "number" && thisX.type == "number") {
2431.                     prev2.value = Math.round(prev2.value / thisX.value*1000000000)
2432.                     /1000000000;
2433.                     thisArr.splice(i-1, 2);
2434.                     i -= 1;
2435.                 }
2436.             }
2437.         }
2438.     }
2439. }
2440.
2441.
2442. for (var i = 2 + startPos ; i < thisArr.length ; ++i) {
2443.     var prev2 = thisArr[i-2];
2444.     var prev = thisArr[i-1];
2445.     var thisX = thisArr[i];
2446.     if (prev.ist == "operator") {
2447.         if (prev.type == "+") {
2448.             if (thisArr.length <= i+1) {
2449.                 if (prev2.ist == "numb" && thisX.ist == "numb") {
2450.                     if (prev2.type == "number" && thisX.type == "number") {
2451.                         prev2.value = Math.round((prev2.value + thisX.value)*1000000000
2452.                         0)/1000000000;
2453.                         thisArr.splice(i-1, 2);
2454.                         i -= 1;
2455.                     }
2456.                 }
2457.             }
2458.         }
2459.     }
2460. }

```

```

2457.         else if (thisArr[i+1].ist != "power") {
2458.             if (thisArr[i+1].ist == "operator") {
2459.                 if (thisArr[i+1].type != "*" && thisArr[i+1].type != "/") {
2460.                     if (prev2.ist == "numb" && thisX.ist == "numb") {
2461.                         if (prev2.type == "number" && thisX.type == "number") {
2462.                             prev2.value = Math.round((prev2.value + thisX.value)*1
000000000)/1000000000;
2463.                             thisArr.splice(i-1, 2);
2464.                             i -= 1;
2465.                         }
2466.                     }
2467.                 }
2468.             }
2469.             else if (prev2.ist == "numb" && thisX.ist == "numb") {
2470.                 if (prev2.type == "number" && thisX.type == "number") {
2471.                     prev2.value = Math.round((prev2.value + thisX.value)*100000000
0)/1000000000;
2472.                     thisArr.splice(i-1, 2);
2473.                     i -= 1;
2474.                 }
2475.             }
2476.         }
2477.     }
2478.     else if (prev.type == "-") {
2479.         if (thisArr.length <= i+1) {
2480.             if (prev2.ist == "numb" && thisX.ist == "numb") {
2481.                 if (prev2.type == "number" && thisX.type == "number") {
2482.                     prev2.value = Math.round((prev2.value -
thisX.value)*1000000000)/1000000000;
2483.                     thisArr.splice(i-1, 2);
2484.                     i -= 1;
2485.                 }
2486.             }
2487.         }
2488.         else if (thisArr[i+1].ist != "power") {
2489.             if (thisArr[i+1].ist == "operator") {
2490.                 if (thisArr[i+1].type != "*" && thisArr[i+1].type != "/") {
2491.                     if (prev2.ist == "numb" && thisX.ist == "numb") {
2492.                         if (prev2.type == "number" && thisX.type == "number") {
2493.                             prev2.value = Math.round((prev2.value -
thisX.value)*1000000000)/1000000000;
2494.                             thisArr.splice(i-1, 2);
2495.                             i -= 1;
2496.                         }
2497.                     }
2498.                 }
2499.             }
2500.             else if (prev2.ist == "numb" && thisX.ist == "numb") {
2501.                 if (prev2.type == "number" && thisX.type == "number") {
2502.                     prev2.value = Math.round((prev2.value -
thisX.value)*1000000000)/1000000000;
2503.                     thisArr.splice(i-1, 2);
2504.                     i -= 1;
2505.                 }
2506.             }
2507.         }
2508.     }
2509. }
2510.
2511.
2512. for (var k = 0 ; k < 5 ; ++k) {
2513.
2514.
2515.     for (var i = 1 + startPos ; i < thisArr.length ; ++i) {
2516.         var prev = thisArr[i-1];
2517.         var thisX = thisArr[i];
2518.         if (prev.ist == "numb") {
2519.             if (prev.type == "number") {
2520.                 if (thisX.ist == "power") {

```

```

2521.         if (thisX.pow_to.inner.length == 1 && thisX.pow_to.inner[0].ist ==
"numb") {
2522.             if (thisX.pow_to.inner[0].type == "number") {
2523.                 prev.value = Math.round(Math.pow(prev.value, thisX.pow_to.
inner[0].value)*1000000000)/1000000000;
2524.                 thisArr.splice(i, 1);
2525.                 i -= 1;
2526.             }
2527.         }
2528.     }
2529. }
2530. }
2531. }
2532.
2533. for (var i = 1 + startPos ; i < thisArr.length ; ++i) {
2534.     var prev = thisArr[i-1];
2535.     var thisX = thisArr[i];
2536.     var num0 = new numb("number", 0);
2537.     var num1 = new numb("number", 1);
2538.
2539.     if (thisX.ist == "power" && thisX.pow_to.inner.length == 1 && thisX.pow_to.inn
er[0].ist == "numb" &&thisX.pow_to.inner[0].type == "number" && thisX.pow_to.inner[0].value == 0) {
2540.         if (prev.ist != "power") {
2541.             thisArr[i-1] = copyThis(num1);
2542.             thisArr.splice(i, 1);
2543.         }
2544.         else {
2545.             thisArr.splice(i-1, 1);
2546.         }
2547.         --i;
2548.     }
2549.     else if (thisX.ist == "power" && thisX.pow_to.inner.length == 1 && thisX.pow_t
o.inner[0].ist == "numb" &&thisX.pow_to.inner[0].type == "number" && thisX.pow_to.inner[0].value == 1)
    {
2550.         thisArr.splice(i, 1);
2551.         --i;
2552.     }
2553.
2554.     prev = thisArr[i-1];
2555.     thisX = thisArr[i];
2556.
2557.     if (thisX.ist == "power" && prev.ist == "numb" && prev.type == "number" && (pr
ev.value == 0 || prev.value == 1)) {
2558.         thisArr.splice(i, 1);
2559.         --i;
2560.     }
2561. }
2562. }
2563.
2564. }
2565.
2566.
2567.
2568.
2569. for (var i = 2 + startPos ; i < thisArr.length ; ++i) {
2570.     var prev2 = thisArr[i-2];
2571.     var prev = thisArr[i-1];
2572.     var thisX = thisArr[i];
2573.
2574.
2575.     if (prev2.ist == "numb") {
2576.         if (prev2.type == "number") {
2577.             if (prev2.value == 0) {
2578.                 if (prev.ist == "operator") {
2579.                     if (prev.type == "+") {
2580.                         thisArr.splice(i-2, 2);
2581.                         i -= 1;
2582.                     }
2583.                     else if (prev.type == "-") {

```

```

2584.         prev2.value = -1;
2585.         prev.type = "*";
2586.     }
2587.     else if (prev.type == "*") {
2588.         thisArr.splice(i-1, 2);
2589.         i -= 1;
2590.     }
2591.     else if (prev.type == "/") {
2592.         thisArr.splice(i-1, 2);
2593.         i -= 1;
2594.     }
2595. }
2596. }
2597. else if (prev2.value == 1) {
2598.     if (prev.ist == "operator") {
2599.         if (prev.type == "*") {
2600.             thisArr.splice(i-2, 2);
2601.             i -= 1;
2602.         }
2603.     }
2604. }
2605. }
2606. }
2607. }
2608.
2609.
2610. for (var i = 2 + startPos ; i < thisArr.length ; ++i) {
2611.     var prev2 = thisArr[i-2];
2612.     var prev = thisArr[i-1];
2613.     var thisX = thisArr[i];
2614.
2615.
2616.     if (thisX.ist == "numb") {
2617.         if (thisX.type == "number") {
2618.             if (thisX.value == 0) {
2619.                 if (prev.ist == "operator") {
2620.                     if (prev.type == "+") {
2621.                         thisArr.splice(i-1, 2);
2622.                         i -= 1;
2623.                     }
2624.                     else if (prev.type == "-") {
2625.                         thisArr.splice(i-1, 2);
2626.                         i -= 1;
2627.                     }
2628.                     else if (prev.type == "*") {
2629.                         if (i <= 2) {
2630.                             thisArr.splice(i-2, 2);
2631.                             i -= 1;
2632.                         }
2633.                         else if (thisArr[i-3].ist != "operator" && thisArr[i-
2634. 2].ist != "power") {
2635.                             thisArr.splice(i-2, 2);
2636.                             i -= 1;
2637.                         }
2638.                         else if (thisArr[i-3].ist == "operator" && thisArr[i-
2639. 3].type != "/" && thisArr[i-2].ist != "power") {
2640.                             thisArr.splice(i-2, 2);
2641.                             i -= 1;
2642.                         }
2643.                     }
2644.                 }
2645.             }
2646.         }
2647.     }
2648.     else if (thisX.value == 1) {
2649.         if (prev.ist == "operator") {
2650.             if (prev.type == "*") {
2651.                 if (i <= 2) {
2652.                     thisArr.splice(i-2, 2);
2653.                     i -= 1;
2654.                 }
2655.                 else if (thisArr[i-3].ist != "operator" && thisArr[i-
2656. 2].ist != "power") {
2657.                     thisArr.splice(i-2, 2);
2658.                     i -= 1;
2659.                 }
2660.                 else if (thisArr[i-3].ist == "operator" && thisArr[i-
2661. 3].type != "/" && thisArr[i-2].ist != "power") {
2662.                     thisArr.splice(i-2, 2);
2663.                     i -= 1;
2664.                 }
2665.             }
2666.         }
2667.     }
2668. }

```



```

2651.         else if (thisArr[i-3].ist != "operator") {
2652.             thisArr.splice(i-1, 2);
2653.             i -= 1;
2654.         }
2655.         else if (thisArr[i-3].type != "/") {
2656.             thisArr.splice(i-1, 2);
2657.             i -= 1;
2658.         }
2659.     }
2660. }
2661. }
2662. }
2663. }
2664. }
2665. }
2666.
2667.
2668.
2669.     for (var k = 0 ; k < 5 ; ++k) {
2670.
2671.         for (var i = 1 + startPos ; i < thisArr.length ; ++i) {
2672.             var prev = thisArr[i-1];
2673.             var thisX = thisArr[i];
2674.             if (prev.ist == "numb") {
2675.                 if (prev.type == "number") {
2676.                     if (thisX.ist == "power") {
2677.                         if (thisX.pow_to.inner.length == 1 && thisX.pow_to.inner[0].ist ==
"numb") {
2678.                             if (thisX.pow_to.inner[0].type == "number") {
2679.                                 prev.value = Math.round(Math.pow(prev.value, thisX.pow_to.
inner[0].value)*1000000000)/1000000000;
2680.                                 thisArr.splice(i, 1);
2681.                                 i -= 1;
2682.                             }
2683.                         }
2684.                     }
2685.                 }
2686.             }
2687.         }
2688.
2689.         for (var i = 1 + startPos ; i < thisArr.length ; ++i) {
2690.             var prev = thisArr[i-1];
2691.             var thisX = thisArr[i];
2692.             var num0 = new numb("number", 0);
2693.             var num1 = new numb("number", 1);
2694.
2695.             if (thisX.ist == "power" && thisX.pow_to.inner.length == 1 && thisX.pow_to.inn
er[0].ist == "numb" && thisX.pow_to.inner[0].type == "number" && thisX.pow_to.inner[0].value == 0) {
2696.                 if (prev.ist != "power") {
2697.                     thisArr[i-1] = copyThis(num1);
2698.                     thisArr.splice(i, 1);
2699.                 }
2700.                 else {
2701.                     thisArr.splice(i-1, 1);
2702.                 }
2703.                 --i;
2704.             }
2705.             else if (thisX.ist == "power" && thisX.pow_to.inner.length == 1 && thisX.pow_t
o.inner[0].ist == "numb" && thisX.pow_to.inner[0].type == "number" && thisX.pow_to.inner[0].value == 1)
{
2706.                 thisArr.splice(i, 1);
2707.                 --i;
2708.             }
2709.
2710.             prev = thisArr[i-1];
2711.             thisX = thisArr[i];
2712.
2713.             if (thisX.ist == "power" && prev.ist == "numb" && prev.type == "number" && (pr
ev.value == 0 || prev.value == 1)) {

```

```

2714.         thisArr.splice(i, 1);
2715.         --i;
2716.     }
2717.
2718. }
2719.
2720. }
2721.
2722.
2723. for (var i = 1 + startPos ; i < thisArr.length ; ++i) {
2724.     var prev = thisArr[i-1];
2725.     var thisX = thisArr[i];
2726.     if (prev.ist == "operator") {
2727.         if (prev.type == "+") {
2728.             if (thisX.ist == "numb") {
2729.                 if (thisX.type == "number") {
2730.                     if (thisX.value < 0) {
2731.                         prev.type = "-";
2732.                         thisX.value *= -1;
2733.                     }
2734.                 }
2735.             }
2736.         }
2737.         else if (prev.type == "-") {
2738.             if (thisX.ist == "numb") {
2739.                 if (thisX.type == "number") {
2740.                     if (thisX.value < 0) {
2741.                         prev.type = "+";
2742.                         thisX.value *= -1;
2743.                     }
2744.                 }
2745.             }
2746.         }
2747.     }
2748. }
2749.
2750.
2751.
2752. for (var i = startPos ; i < thisArr.length ; ++i) {
2753.     if (thisArr[i].ist == "function") {
2754.         thisArr[i].inner.inner = optimize(thisArr[i].inner.inner, 0);
2755.     }
2756. }
2757.
2758. for (var i = startPos ; i < thisArr.length ; ++i) {
2759.     if (thisArr[i].ist == "power") {
2760.         thisArr[i].pow_to.inner = optimize(thisArr[i].pow_to.inner, 0);
2761.     }
2762. }
2763.
2764. for (var i = startPos ; i < thisArr.length ; ++i) {
2765.     if (thisArr[i].ist == "function" && thisArr[i].type == "bracket") {
2766.         if (thisArr[i].inner.inner.length == 1) {
2767.             var outerNew;
2768.             if (thisArr[i].inner.inner[0].ist == "function") {
2769.                 outerNew = thisArr[i].inner.outer;
2770.                 thisArr[i].inner.inner[0].inner.outer = thisArr[i].inner.outer;
2771.             }
2772.             thisArr[i] = thisArr[i].inner.inner[0];
2773.             if (thisArr[i].ist == "function") {
2774.                 thisArr[i].inner.outer = outerNew;
2775.             }
2776.         }
2777.     }
2778. }
2779.
2780. for (var i = startPos ; i < thisArr.length ; ++i) {
2781.     if (thisArr[i].ist == "function" && thisArr[i].type == "ln") {

```

```

2782.         if (thisArr[i].inner.inner.length == 1 && thisArr[i].inner.inner[0].ist == "nu
mb" && thisArr[i].inner.inner[0].type == "e") {
2783.             thisArr[i] = new numb("number", 1);
2784.         }
2785.         else if (thisArr[i].inner.inner.length == 1 && thisArr[i].inner.inner[0].ist =
= "numb" && thisArr[i].inner.inner[0].type == "number" && thisArr[i].inner.inner[0].value == 1) {
2786.             thisArr[i] = new numb("number", 0);
2787.         }
2788.     }
2789. }
2790.
2791.     return thisArr;
2792. }
2793.
2794. this.prestigeB.addEventListener("click", pviss.bind(this));
2795. function pviss() {
2796.     if (!pVis) {
2797.         pVis = true;
2798.     }
2799.     else pVis = false;
2800. }
2801.
2802.
2803. function sign(x) {
2804.     return (x>0 ? 1 : -1);
2805. }
2806.
2807. function short(num) {
2808.     if (num < 1000) {
2809.         return Math.round(num);
2810.     }
2811.     else if (num < 1e6) {
2812.         return Math.round(num/100)/10 + " K";
2813.     }
2814.     else if (num < 1e9) {
2815.         return Math.round(num/1e5)/10 + " M";
2816.     }
2817.     else if (num < 1e12) {
2818.         return Math.round(num/1e8)/10 + " B";
2819.     }
2820.     else if (num < 1e15) {
2821.         return Math.round(num/1e11)/10 + " T";
2822.     }
2823.     else if (num < 1e18) {
2824.         return Math.round(num/1e14)/10 + " Qa";
2825.     }
2826.     else if (num < 1e21) {
2827.         return Math.round(num/1e17)/10 + " Qi";
2828.     }
2829.     else if (num < 1e24) {
2830.         return Math.round(num/1e20)/10 + " Sx";
2831.     }
2832.     else if (num < 1e27) {
2833.         return Math.round(num/1e23)/10 + " Sp";
2834.     }
2835.     else if (num < 1e30) {
2836.         return Math.round(num/1e26)/10 + " Oc";
2837.     }
2838.     else if (num < 1e33) {
2839.         return Math.round(num/1e29)/10 + " No";
2840.     }
2841.     else if (num < 1e36) {
2842.         return Math.round(num/1e32)/10 + " Dc";
2843.     }
2844.     else {
2845.         var numPow = Math.floor((Math.log(num) / Math.log(10)))
2846.         return Math.round(num/Math.pow(10,numPow-1))/10 + "e" + numPow;
2847.     }
2848. }

```