



PROJECT REPORT

GRAPH PLOTTER

LAB- BATCH-331

Submitted to-

D. B. Phatak
(Chair Professor, CS Department
IIT Bombay)



ACKNOWLEDGEMENTS:

This whole journey of “graph plotting” was a great learning experience.

This learning was not only the familiarity of codes but was full with plethora of teamwork, professionalism, leadership, working under time constraints.

The help and support given by our T.A PEEYUSH boosted the progress of our project.

And finally grand accolades to Dr. Deepak .B. Phatak for organizing such enthusiastic and a wonderful project work!!!

Team:A

Gaurav Chauhan (Coordinator)

Himanshu Kumar Bharti

Gourav Basu

Team:B

Harvinder Singh Gill

Sriraj Eluri

Hansraj Kumawat

Vishnu



INTRODUCTION

Our project is doing the project of 2D Graph Plotter. It also includes calculating derivative, integration, critical points and roots of the function given by user.

SALIENT FEATURES OF THE PROJECT:

- Graph of function
- Graph of Derivative
- Graph of one indefinite integral
- Calculation of roots of the function
- Displaying the location of critical points
- Movement of display window over the different regions of xy plane.



This will take the following functions:

Function	:	Syntax
Sine x	:	$\sin(x)$
Cosine x	:	$\cos(x)$
Tangent x	:	$\tan(x)$
Secant x	:	$1/\cos(x)$
Cosecine x	:	$1/\sin(x)$
Cotangent x	:	$1/\tan(x)$
Arc sine	:	$\text{asin}(x)$
Arc cosine	:	$\text{acos}(x)$
Arc tangent	:	$\text{atan}(x)$
Exponential	:	$\exp(x)$
Natural logarithm	:	$\log(x)$
Common logarithm	:	$\log_{10}(x)$
Power	:	$\text{base}^{\text{exponent}}$



User Interface Requirements

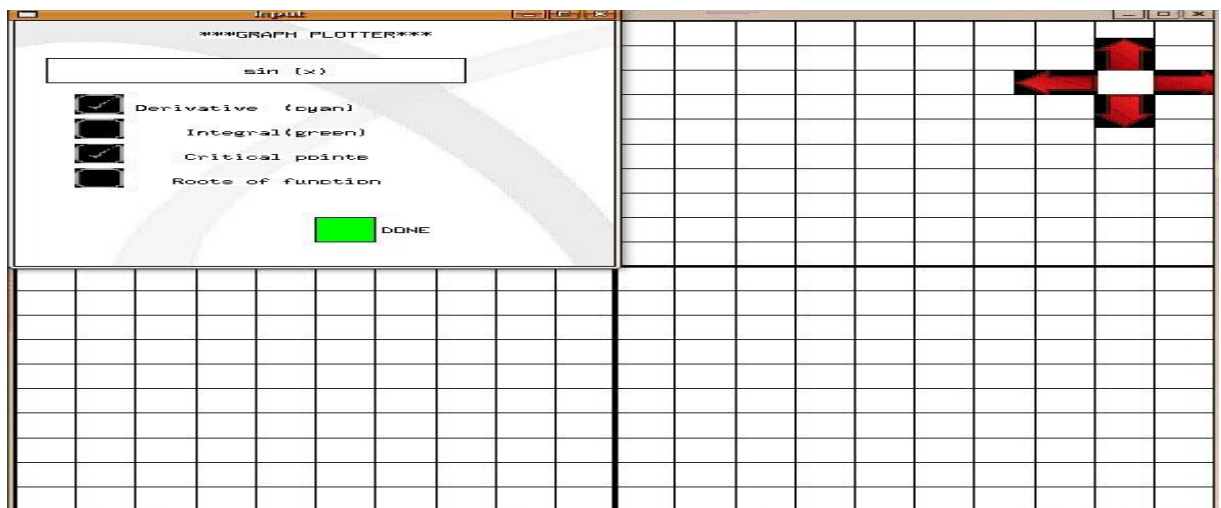
Software Requirements:

- Calculator (downloaded from www.speqmath.com)
- EzWindows package

How to execute:

Graph plotter takes the input of function from the user in the terminal window. Then the user is prompted to select the various option of plotting the graphs of derivative, integral etc. besides the graph of function.

The “input Window” with the various option:





There are tick boxes present in the window like as given below:



After clicking on this it appears like this:



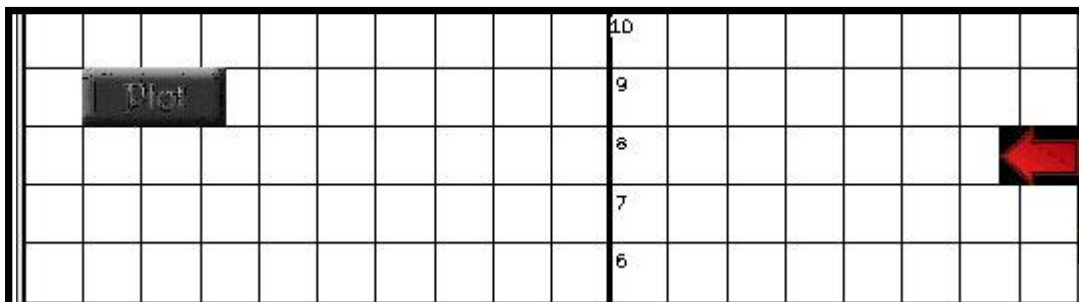
Tick boxes will enable the function and plot its graph.

Press “Done” key to continue.

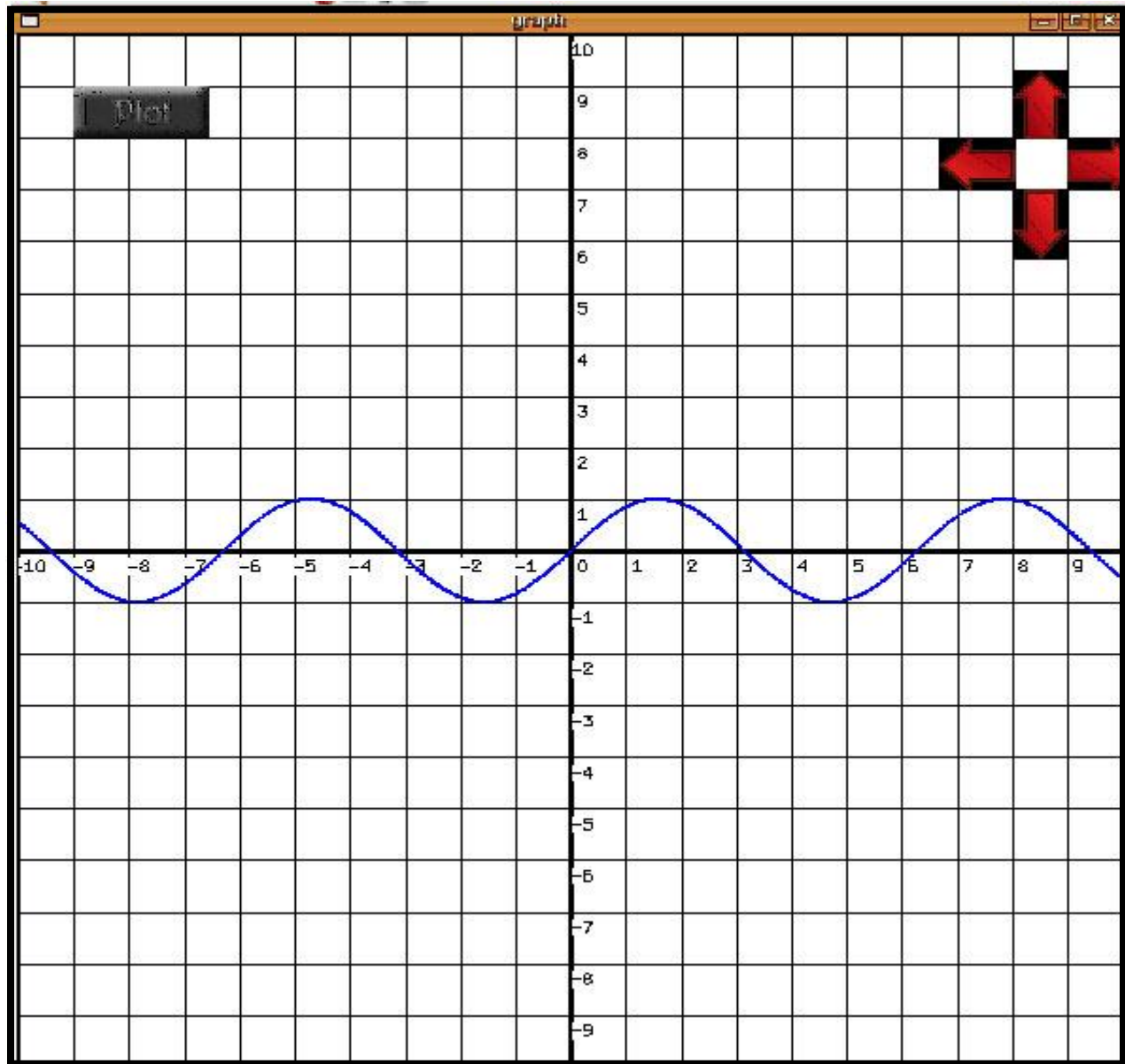


DONE

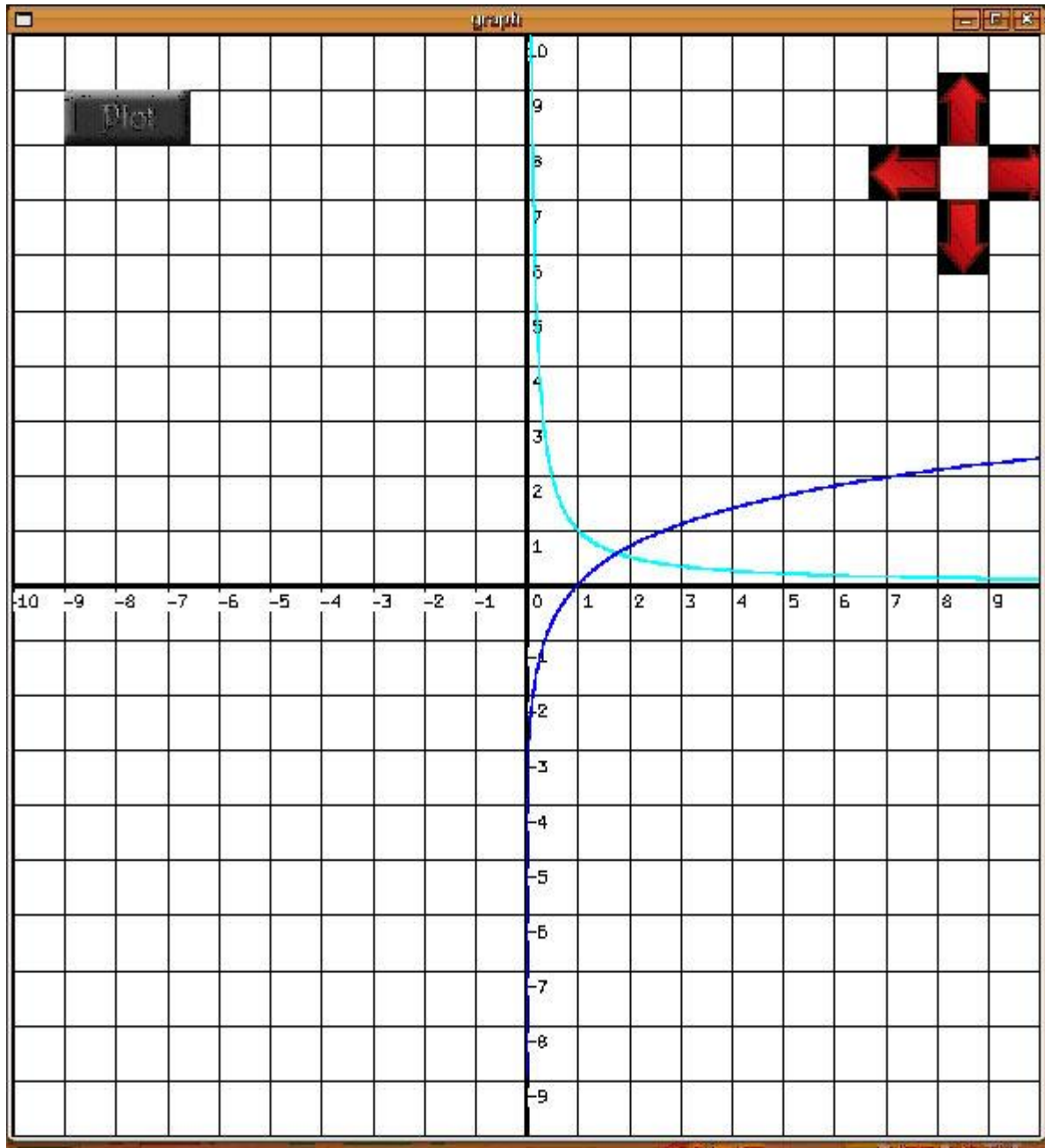
Now to display the graph click on the upper left corner (Plot)



The graph of function is displayed in a window.



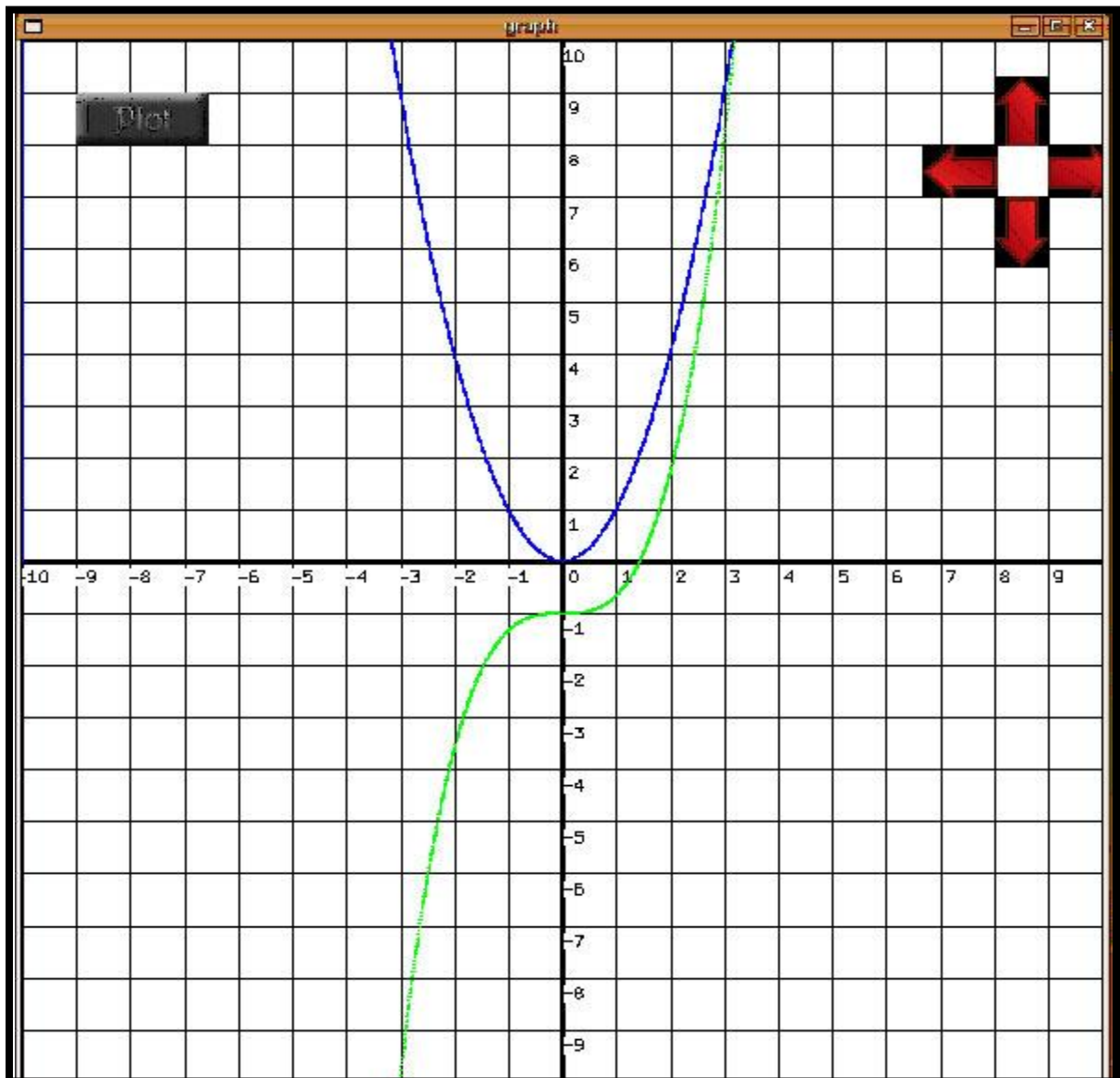
The graph of $\log(x)$ (blue) along with its derivative (cyan)





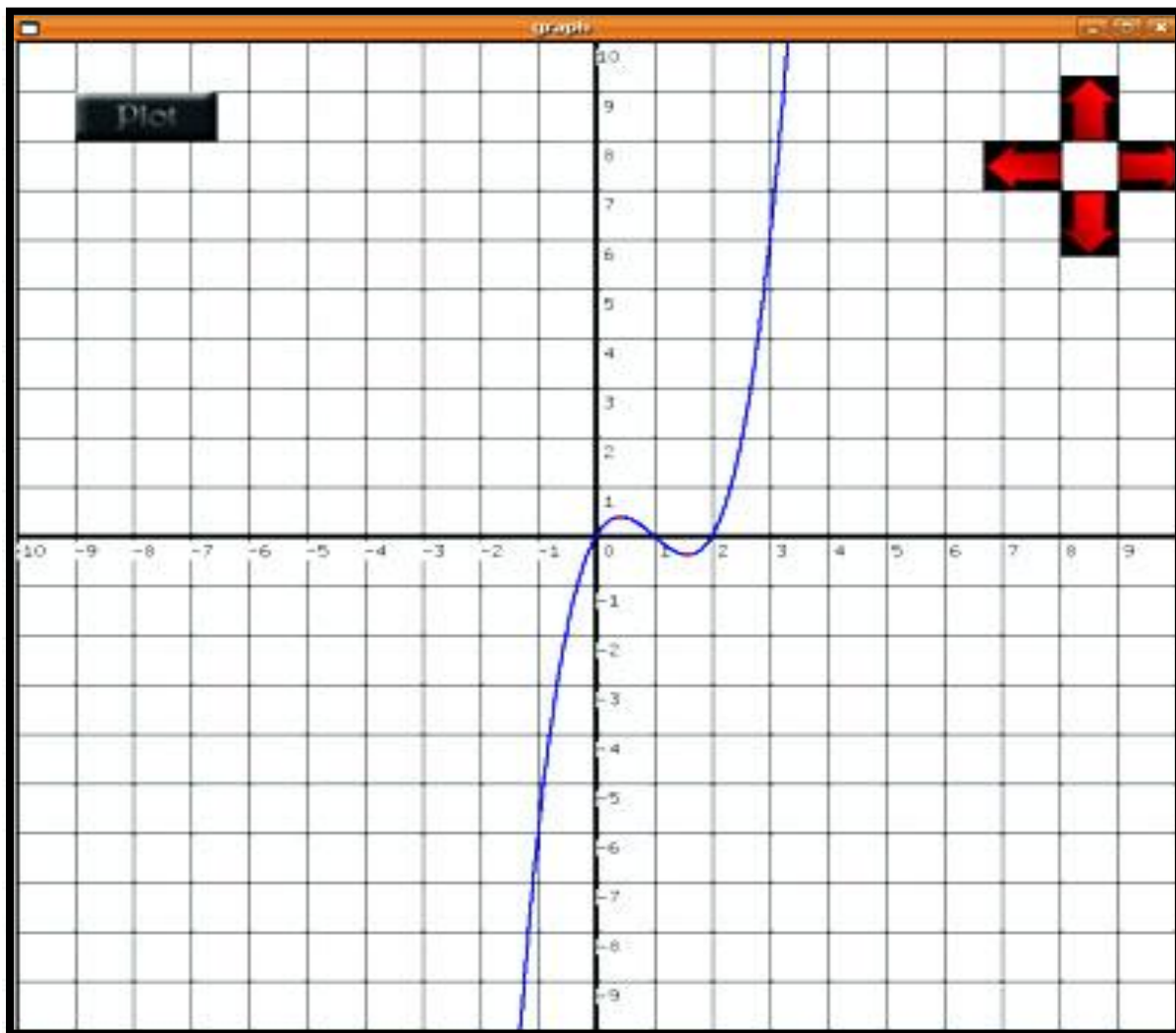
The constant 'C' in the indefinite integral is taken in such a way that the integral passes through point $(x,y)=(0,-1)$.

The graph of x^2 along with its integral.



The critical points of the graph are calculated and marked in red colour.

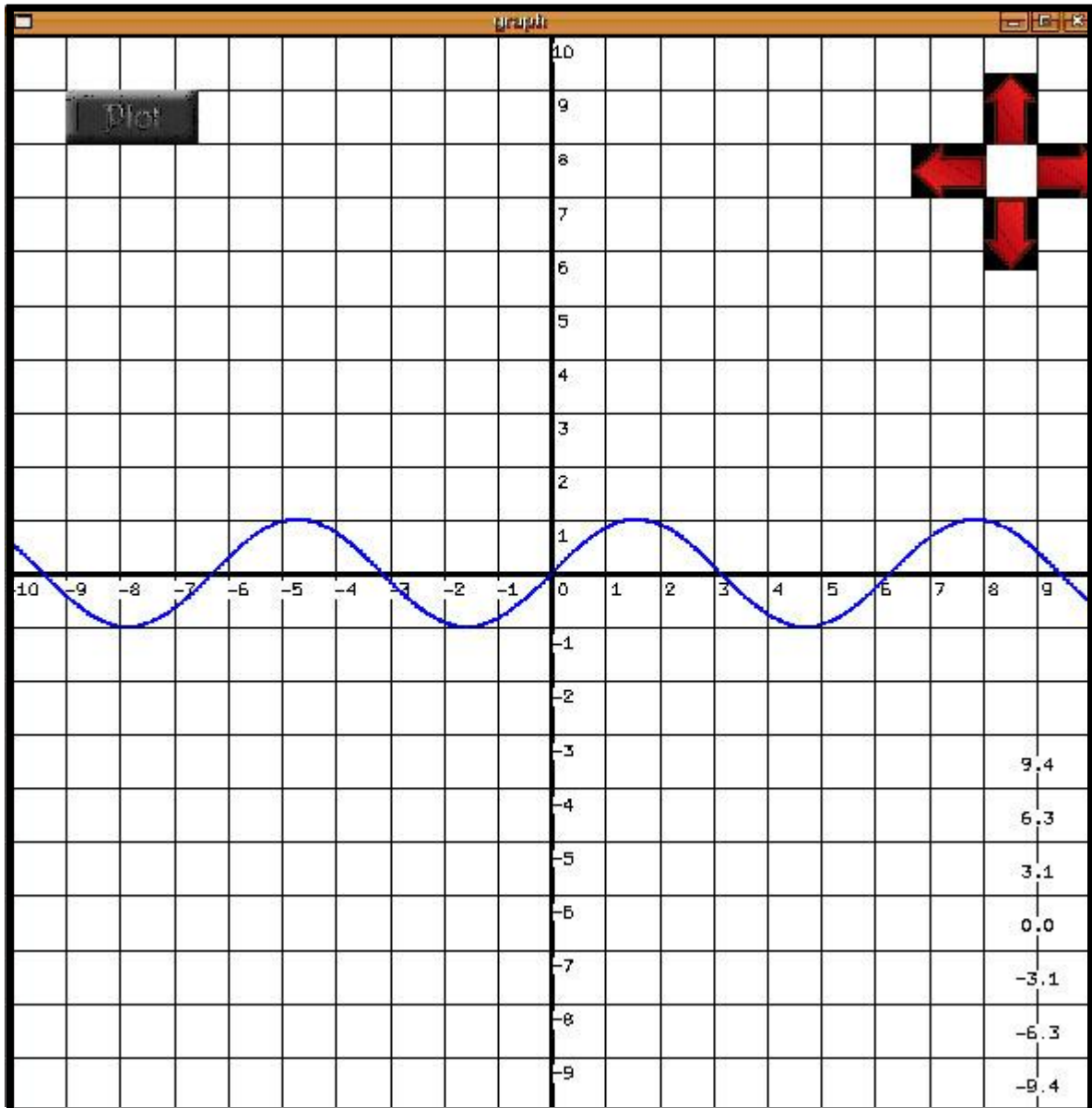
$$Y=(x-1)*(x-2)*x$$





All the roots (if any) of the function are displayed on the lower right corner.

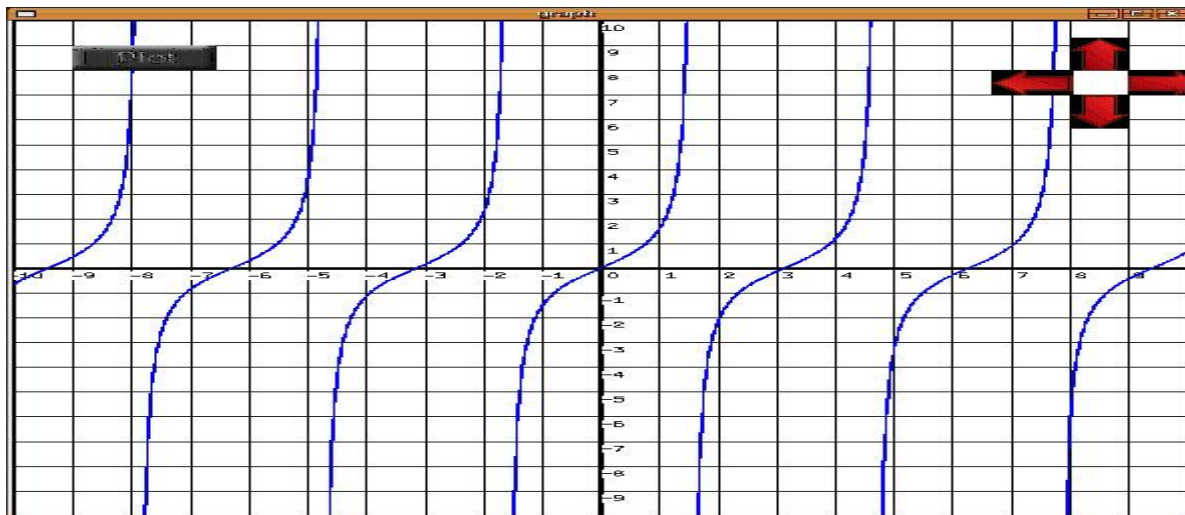
The roots of $\sin(x)$ are displayed in the following fig.



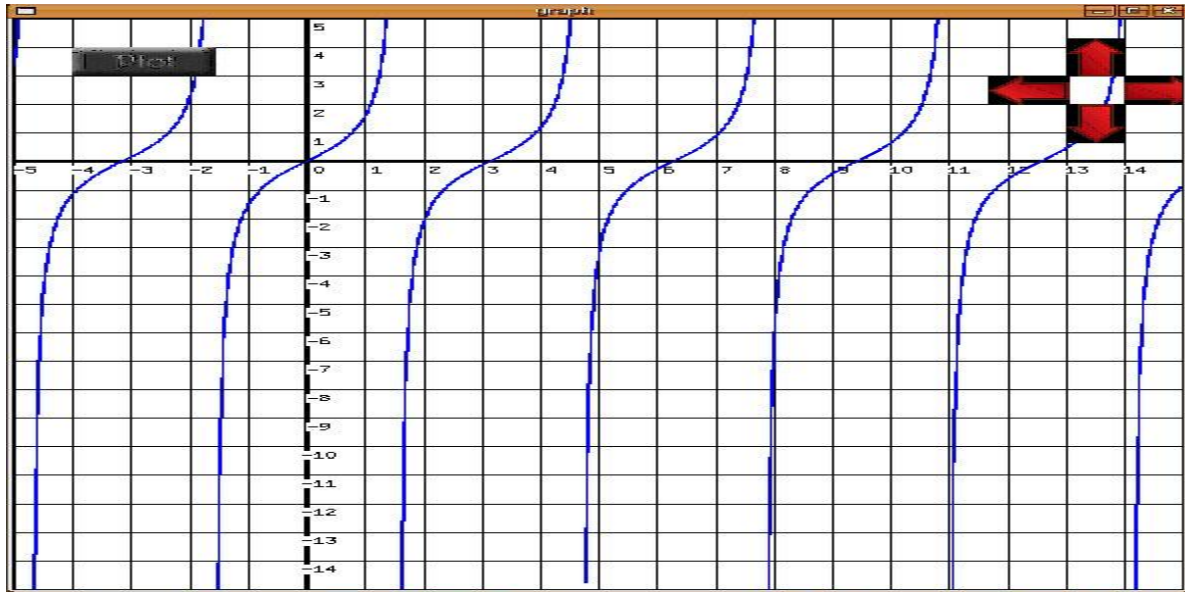
By default the graph is drawn between $X=(-10,10)$, $Y=(-10,10)$. But user is given the option to maneuver the graph plane with the help of arrow keys displayed on the top right corner.

The various position of graph is displayed in the following fig.

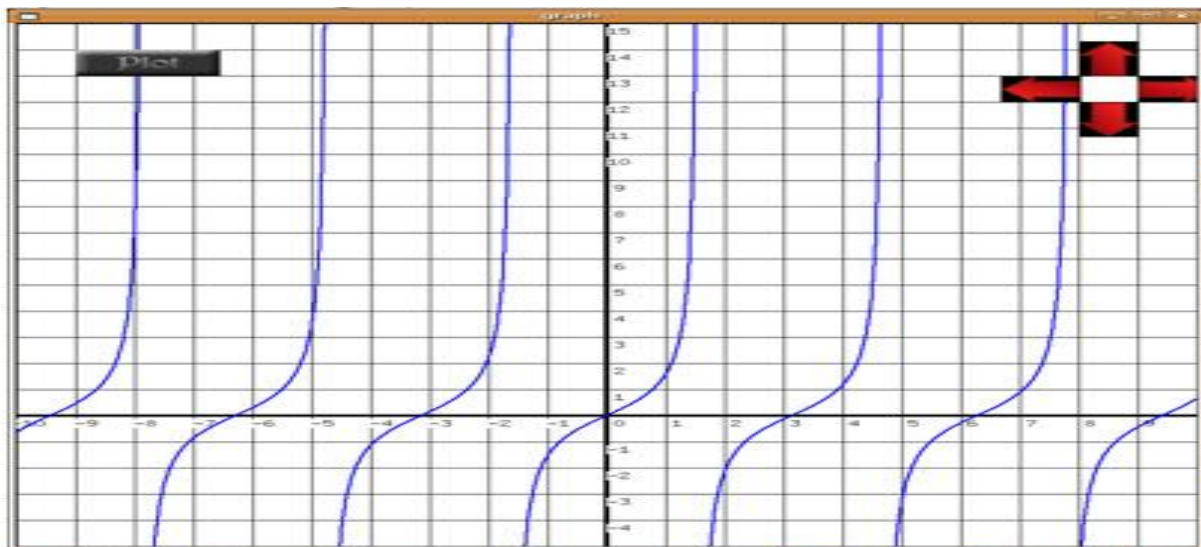
Default screen:



After pressing up key and left key once:



After pressing down key once:





Plan :

After a brief online survey & consultation with our TA, the nature of the project & its requirements were understood. The core team of seven members was divided into two teams A & B

Team A was assigned the task of handling the input of a function from the user and handle it with parser.h to pass on the x,y pairs to the output.

Team B was assigned the task of handling the output through the EZ-Windows and display the graphs asked by the user.



ERROR HANDLING:

This program is also able to detect errors (if any) in the entered function. It does it by comparing the stored data present in it to the wrongly entered string and then accordingly returns the statement.

```
{
switch (id)
{
    // syntax errors
    case 1: return "Syntax error in part \"%s\"";
    case 2: return "Syntax error";
    case 3: return "Parenthesis ) missing";
    case 4: return "Empty expression";
    case 5: return "Unexpected part \"%s\"";
    case 6: return "Unexpected end of expression";
    case 7: return "Value expected";

    // wrong or unknown operators, functions, variables
    case 101: return "Unknown operator %s";
    case 102: return "Unknown function %s";
    case 103: return "Unknown variable %s";

    // domain errors
    case 200: return "Too long expression, maximum number of characters
exceeded";
    // error in assignments of variables
    case 300: return "Defining variable failed";

    // error in functions
    case 400: return "Integer value expected in function %s";
    }
    return "Unknown error";
}
```

The above is a part of the main C++ error detecting program.

It can also tell the column and row number in which the error is found.



Code Architecture

Input is handled by following two functions:

```
/******INPUT FUNCTIONS******/  
/******replaces x with a num value (parser)***** */
```

```
void convert(char *str,char *rep,char *newstr)
```

```
{  
    char ch;  
    ch='x';  
    int i,j,k;  
        i=k=0;  
    while (str[i] !='\0')  
    {  
        if(str[i]==ch)  
            for(j=0;j<strlen(rep);j++)  
                newstr[k++]=rep[j];  
        else newstr[k++]=str[i];  
        i++;  
    }  
    newstr[k]='\0';  
}
```

```
/****** parser function ******/
```

```
float gh(float x)
```

```
{  
    char arbid[56];  
    char err[255],newstr[200];
```



```
char rep[25];
double res = 0;
float y;
// create a parser object
Parser prs;
sprintf(rep,"%f",x);
convert(str,rep,newstr);

if (strcmp(str, "") != 0)
{
    try
    {
        // evaluate the expression
        char* result;
        result = prs.parse(newstr);

        // error display if a wrong input is added

        if ( *result == 'E') {puts (result); exit (0);}
        sscanf(result,"%s%s%f", arbid,arbid,&y);

    }
    catch (...)
    {
        printf("\tError: Unknown error occurred in parser\n");// when
all is hopeless
    }
}
return y;
}
```



The movement code:

A salient feature of the graph plotter, the movement keys are handled by various variable like (toright, toup, right up) and finally executed through following render line command:

W. RenderLine (Position(xnot,10-y+factor*up), Position(xnot+acc,10-ynot+factor*up), Red,0.1)

The graph is finally displayed by following Graph function:

Void graph (int factor,int right,int up,float acc,int iterations)

{

float ynot, k=0.0, xnot = 0.0,y,x;

int i;

x = right*factor - 10;

ynot = gh(x);

for (i=0 ; i<iterations;i++)

{

 y = ynot;

 x += acc;

 ynot = gh(x);

 if (y < 12+factor*up&& y> -12 +factor*up){

 w.RenderLine (Position(xnot,10-y+factor*up),Position(xnot+acc,10-ynot+factor*up),Blue,0.05);}



```
    xnot +=acc;
}
if ( ifCritical == 1){
graphWithCriticalpoints ( factor,right, up, acc, iterations);}
if ( ifRoot == 1) {
graphWithRoots ( factor, right,up, acc, iterations);}
}
```

Limitations:

- Can handle only explicit functions
- Follows strict syntax style which user has to memories
- Only specific indefinite integral can be drawn.
- As the graph is calculated on the point wise basis, the graph may skip some such points or add a few extra when the slope of any graph approaches



References

- www.speqmath.com
- C++ Design by Cohoon and Davidson.

Credits

- Juha Nieminen and Joel Yliluoma (for development of fparser library).
- Our TA: Peeyush Gupta