## Math lib

int i;

## Absolute value function

##int abs(int x) :

## return (x < 0) ? -1\*x : x;

##:

##exponential function

float exp(float x\_) :

float result\_ = 1.0;

float term\_ = 1.0;

for i = 1; i <= 20; i++: {

term\_ \*= x\_ / i;

result\_ += term\_;

}

return result\_;

:

## Power function (base^exponent)

float power(float base, float exponent) :

float result = 1.0;

if exponent > 0: {

for i = 0; i < exponent; ++i: {

result \*= base;

}

} else if exponent < 0: {

for i = 0; i < -1\*exponent; ++i: {

result \*= base;

}

result = 1.0 / result;

}

return result;

:

int mod(int x, int y) :

if y == 0 :{

return -1;

}

int quotient = x / y;

int gg = x - (quotient \* y);

if y > 0 && gg < 0: {

gg += y;

}

if y < 0 && result > 0: {

gg += y;

}

return gg;

:

## GCD function (Greatest Common Divisor using Euclidean algorithm)

int gcd(int a, int b) :

while b != 0: {

int temp = b;

b = a % b;

a = temp;

}

return a;

:



## ARRAY\_LIB

##ifndef MYARRAYLIB\_H

##define MYARRAYLIB\_H

##include <iostream>

## Global variable for iteration (used to avoid reusing variable names)

##extern int global\_iter;

int global\_iter;

## Sort function (Bubble Sort) for an array of integers

void my\_sort(int$ arr, int size) :

bool swapped = true;

while swapped: {

swapped = false;

for global\_iter = 0; global\_iter < size - 1; ++global\_iter: {

if arr[global\_iter] > arr[global\_iter + 1]: {

## Swap if elements are out of order

int temp = arr[global\_iter];

arr[global\_iter] = arr[global\_iter + 1];

arr[global\_iter + 1] = temp;

swapped = true; ## If a swap is made, continue checking

}

}

}

:

## Function to find the minimum element in the array

int my\_min(int$ arr1, int size2) :

if size2 <= 0: {

return -1; ## Error: empty arr1ay

}

int min\_val = arr1[0]; ## Assume the first element is the minimum

for global\_iter = 1; global\_iter < size2; ++global\_iter: {

if arr1[global\_iter] < min\_val: {

min\_val = arr1[global\_iter]; ## Update if a smaller element is found

}

}

return min\_val;

:

## Function to find the maximum element in the arr2ay

int my\_max(int$ arr2, int size3) :

if size3 <= 0: {

return -1; ## Error: empty arr2ay

}

int max\_val = arr2[0]; ## Assume the first element is the maximum

for global\_iter = 1; global\_iter < size3; ++global\_iter: {

if arr2[global\_iter] > max\_val: {

max\_val = arr2[global\_iter]; ## Update if a larger element is found

}

}

return max\_val;

:



## STRING\_LIB



## Function to calculate the length of a string

int strlen(char $strlenstr)

:

int length = 0;

while strlenstr[length] != '\0' :

{

length++;

}

return length;

:

## Function to copy a string from src to dest

void strcpy(char $dest, char $src)

:

int i = 0;

while src[i] != '\0' :

{

dest[i] = src[i];

i++;

}

dest[i] = '\0'; ## Null terminate the destination string

:

## Function to compare two strings (similar to strcmp)

int strcmp(char $str1, char $str2)

:

while $str1 != '\0' && $str2 != '\0' :

{

if $str1 != $str2 :

{

return ($str1 - $str2);

}

str1++;

str2++;

}

return ($str1 - $str2); ## If both strings are equal till the null-terminator

:

## Function to concatenate two strings (similar to strcat)

void strcat(char $dest, char $src)

:

while $dest != '\0':

{

dest++; ## Move to the end of the destination string

}

while $src != '\0':

{

$dest = $src; ## Copy each character from src to dest

dest++;

src++;

}

$dest = '\0'; ## Null terminate the concatenated string

:

## Function to convert a string to uppercase

void to\_uppercase(char $strToUpper)

:

while $strToUpper != '\0':

{

if $strToUpper >= 'a' && $strToUpper <= 'z' :

{

$strToUpper = $strToUpper - 'a' + 'A'; ## Convert to uppercase

}

strToUpper++;

}

:

## Function to convert a string to lowercase

void to\_lowercase(char $strToLower)

:

while $strToLower != '\0':

{

if $strToLower >= 'A' && $strToLower <= 'Z' :

{

$strToLower = $strToLower - 'A' + 'a'; ## Convert to lowercase

}

strToLower++;

}

:

## Function to check if a string is a palindrome

int is\_palindrome(char $strIsPalindrome)

:

int length = strlen(strIsPalindrome);

int i = 0;

while i < length / 2:

{

if strIsPalindrome[i] != strIsPalindrome[length - i - 1] :

{

return 0; ## Not a palindrome

}

i++;

}

return 1; ## Is a palindrome

:

## Function to reverse a string

void reverse(char $strReverse)

:

int length = strlen(strReverse);

int i = 0;

while i < length / 2:

{

char temp = strReverse[i];

strReverse[i] = strReverse[length - i - 1];

strReverse[length - i - 1] = temp;

i++;

}

:

## Function to convert a string to an integer

int atoi(char $strAtoi)

:

int num = 0;

int sign = 1;

if $strAtoi == '-' :

{

sign = -1;

strAtoi++;

}

while $strAtoi != '\0':

{

num = num \* 10 + ($strAtoi - '0');

strAtoi++;

}

return num \* sign;

:

## Function to convert an integer to a string

void itoa(int num, char $strItoa)

:

int i = 0;

int is\_negative = 0;

if num < 0 :

{

is\_negative = 1;

num = -num;

}

while num != 0:

{

strItoa[i++] = num % 10 + '0';

num = num / 10;

}

if is\_negative :

{

str[i++] = '-';

}

strItoa[i] = '\0';

reverse(strItoa);

:



## vectorLibrary

## Initializes a vector with a given value

void vector\_init(float$ vec\_init, int vec\_size, float init\_value) :

int init\_i = 0;

while init\_i < vec\_size: {

vec\_init[init\_i] = init\_value;

init\_i++;

}

:

## Adds two vectors element-wise

void vector\_add(float$ vec\_result, float$ vec\_a, float$ vec\_b, int vec\_len) :

int add\_i = 0;

while add\_i < vec\_len: {

vec\_result[add\_i] = vec\_a[add\_i] + vec\_b[add\_i];

add\_i++;

}

:

## Multiplies a vector by a scalar value

void vector\_scale(float$ scale\_vec, int scale\_size, float scale\_factor) :

int scale\_i = 0;

while scale\_i < scale\_size: {

scale\_vec[scale\_i] \*= scale\_factor;

scale\_i++;

}

:

## Computes the dot product of two vectors

float vector\_dot(float$ dot\_vec1, float$ dot\_vec2, int dot\_size) :

float dot\_result = 0.0;

int dot\_i = 0;

while dot\_i < dot\_size: {

dot\_result += dot\_vec1[dot\_i] \* dot\_vec2[dot\_i];

dot\_i++;

}

return dot\_result;

:



# Primitive\_Test\_Libs

## Screen\_lib



/\*\*

\* Graphic screen library.

\*/

class Screen :

Array screen;

boolean color; ##true for black, false for white

void init() :

int temp1;

temp1 = Math.twoToThe(12); ## 2^12 = 4096

temp1 = 34\*temp1; ## 34\*4096 = 139264

temp1 = temp1 + 1408; ## 139264 + 1408 = 140672

screen = 0-temp1; ## 4-140672 = -140668 //since screen[0] not working

color = true;

:

void clearScreen() :

int i,j;

i = 0;

while i < 2400: {

j=i+i+i+i;

screen[j] = 0;

i = i+1;

}

:

/\*\* Sets the color to be used in further draw commands

\* where white = false, black = true. \*/

:;



## Iostream\_lib

## Outputs a str to the console

void printString(str message):

Output.printString(message);

return;

:

## Outputs a character to the console

void printChar(char c):

Output.printChar(c);

return;

:

## Outputs an integer to the console

void printInt(int value):

Output.printInt(value);

return;

:

## Reads a line of input from the console

str readLine():

str line;

line = Keyboard.readLine("");

return line;

:

## Reads an integer from the console

int readInt():

int value;

value = Keyboard.readInt("");

return value;

:

## Overloaded insertion operator equivalent (<<)

void operator<< (str message):

Iostream.printString(message);

return;

:

void operator<< (int value):

Iostream.printInt(value);

return;

:

void operator<< (char c):

Iostream.printChar(c);

return;

:

## Overloaded extraction operator equivalent (>>)

void operator>> (int$ value):

$value = Iostream.readInt();

return;

:

void operator>> (String$ line):

$line = Iostream.readLine();

return;

:

## Keyboard\_lib

void Keyboard() :

Array keyboard;

int isCaps;

## Initializes the keyboard.

void init() :

keyboard = 16383;

keyboard = keyboard \*4;

isCaps = 0;

return;

:

char keyPressed() :

return keyboard[0];

:

char readChar() :

char key;

int i;

i=0;

while i = 0: {

key = Keyboard.keyPressed();

if key = 9: {

if isCaps = 0: {

isCaps = 32;

}

else: {

isCaps = 0;

}

}

else: {

if key = 0: {

}

else: {

i=1;

}

}

}

if key >96: {

if key < 123: {

key=key-isCaps;

Output.printChar(key);

}

}

if key > 47: {

if key<58: {

Output.printChar(key);

}

}

return key;

:

str readLine(str message) :

str line;

char c;

Output.printstr(message);

line = str.new(50);

c = Keyboard.readChar();

while !(c = str.newLine()): {

if c = str.backSpace():

{

line.eraseLastChar();

Output.backSpace();

}

else: {

line.appendChar(c);

}

c = Keyboard.readChar();

}

return line;

:

int readInt(str message) :

var str line;

line = Keyboard.readLine(message);

return line.intValue();

:

:

