CSCI 610
ADVANCED CONCEPTS IN OPERATING SYSTEMS

Homework 2

Purpose

Distributed Hash tables and Gossiping Protocol

Swapnil Acharya

StarId: ea4963aw

Due Date: 04/06/2020

Problem C

MakeFile

```
all: fTable

fTable: FingerTable.h FingerTable.cpp main.cpp
    g++ FingerTable.h FingerTable.cpp main.cpp -o fTable

clean:
    rm -f fTable
```

CODE

```
/*
 * @author Swapnil Acharya
 * @since 03/21/2020
 */

#ifndef _FINGERTABLE_H_
#define _FINGERTABLE_H_


#include <vector>
using namespace std;

class FingerTable{

    public:
        FingerTable(); //default constructor
        FingerTable(int _id); //constructor that allows id
        FingerTable(int _id, vector< pair<int,int> > _tbl); //constructor that sets  node
is, and finger table
        FingerTable(int _id, vector< pair<int,int> > _tbl, FingerTable* _link);

        //setters
        void setNodeId(int _id); //method to set node id
        void setTable(vector< pair<int,int> > _tbl); //sset finger table node
        void setNextFingerTable(FingerTable* _link); //set next node address in finger
table

        //getters
        int getNodeId() const; //get node IDs of the node
        vector< pair<int,int> > getTable() const; //get the finger table for this node
        FingerTable* getNext() const; //get the next feild table address

        void displayFingerTable();  //display the finget table for this node

    private:
        int _nodeId; //feild to hold id of a node
        vector< pair<int,int> > _table; //feild to hold feinger tble for node
        FingerTable* _next; //filed to hold next addes of the feilder table
};

#endif
```

```
#include "FingerTable.h"
#include <stdio.h>
#include <vector>
#include <iterator>


FingerTable::FingerTable(){
    _nodeId = 0;
    _next = NULL;
}

FingerTable::FingerTable(int _id){
    _nodeId = _id;
    _next = NULL;
}

FingerTable::FingerTable(int _id, vector< pair<int,int> > _tbl){
    _nodeId = _id;
    _table = _tbl;
    _next = NULL;
}

FingerTable::FingerTable(int _id, vector< pair<int,int> > _tbl, FingerTable* _link){
    _nodeId = _id;
    _table = _tbl;
    _next = _link;
}


void FingerTable::setNodeId(int _id){
    _nodeId = _id;
}

void FingerTable::setTable(vector< pair<int,int> > _tbl){
    _table = _tbl;
}

void FingerTable::setNextFingerTable(FingerTable* _link){
    _next = _link;
}



int FingerTable::getNodeId() const{
    return _nodeId;
}

vector< pair<int,int> > FingerTable::getTable() const{
    return _table;
}

FingerTable* FingerTable::getNext() const{
    return _next;
}


void FingerTable::displayFingerTable(){

    std::vector< pair<int,int> >::iterator _ptr;

    printf("Node: %i\n",_nodeId);

     printf("---------------------\n");
```

```
    for(_ptr = _table.begin(); _ptr < _table.end(); _ptr++){
        printf("%i   |  %i\n", (*_ptr).first, (*_ptr).second);

    }
    printf("---------------------\n");
    printf("\n");
}



/*
 * @author Swapnil Acharya
 * @since 4/4/20
 */
#include <stdlib.h>
#include <stdio.h>
#include <vector>
#include <iterator>
#include <algorithm>
#include <math.h>
#include "FingerTable.h"

#define IDENTIFIER_SPACE 19

/*
 * This function return the closeset sucessor to a given node for
 * finger table construction
 * @param _idList NodeList
 * @param _currentSucessor the nodeid whoe sucessor is to be found
 * @return return the sucessor
 */
int getClosetSucessor(std::vector<int> _idList, int _nodeId){


    std::vector<int>::iterator _ptr;
    _ptr = _idList.begin();
    for(_ptr = _idList.begin(); _ptr<_idList.end(); _ptr++){
        if(*_ptr > _nodeId){
            return *_ptr; //return a sucessor than nodeId
     }
    }
    return _idList.front(); //if the node is the last node list it to the first node
}

/*
 * This function generates fingertable for a given nodeId
 * @param _nodeId node if for whose finger table is to be generated
 * @param _idList list of nodeIds
 * @return fingertable
 */
std::vector< pair<int,int> > generateFingerTable(int _nodeId, std::vector<int> _idList){

    std::vector< pair<int,int> > _aTable;

    //generate sucessor
    int i =0;
    int _sucessor = 0;

    for(i = 1; i <= IDENTIFIER_SPACE; i++){

    //get sucessor
    //sucessor = (nodeId + 2^(i-1)) Mod 2^(m)
        _sucessor = 0;
```

```
        _sucessor = _nodeId +  (int)( pow(2,(i-1)) );
        _sucessor = _sucessor % (int)(pow(2,IDENTIFIER_SPACE));

    //if the sucessor is on the node list pussh tat node is the table
        if(binary_search(_idList.begin(), _idList.end(),_sucessor)){
            _sucessor = _sucessor;
        }
    //else get the close sucessor
        else{
            _sucessor = getClosetSucessor(_idList,_sucessor);
        }

        _aTable.push_back(make_pair(i,_sucessor));
    }

    return _aTable; //returnt he generated finger table
}


int main(){

    std::vector<int> _ids; //vector to hold node ids

    //read file
    FILE * _fp;
    _fp = fopen("nodeIDs","r"); //open NodeIds file for read
    int _temp = 0;

    //read nodeIDs from fiel to vector
    while(!feof(_fp)){
        _temp = 0;
        fscanf(_fp,"%i",&_temp);
        _ids.push_back(_temp);
    }
    _ids.pop_back(); //remove the last endline read into vector
    fclose(_fp); //close the file pointer

    //sort vector containing node ids
    sort(_ids.begin(),_ids.end());

    //start generating finger tables
    FingerTable* _head;
    FingerTable* _currPtr;
    std::vector<int>::iterator _ptr;

    //generate finger table for all node ids and put then in acircular linked list
    for(_ptr = _ids.begin(); _ptr <_ids.end(); _ptr++){

        if(*_ptr == _ids.front()){
            std::vector< pair<int,int> > _tempTable = generateFingerTable(*_ptr,_ids);
            FingerTable* _temp = new FingerTable(*_ptr,_tempTable);
            _head = _temp;
            _currPtr = _temp;
            _temp = NULL;
        }
        else{
            std::vector< pair<int,int> > _tempTable = generateFingerTable(*_ptr,_ids);
            FingerTable* _temp = new FingerTable(*_ptr,_tempTable);
            _currPtr->setNextFingerTable(_temp);
            _currPtr = _temp;
            _temp = NULL;
        }
    }
```

```
    //linking last item's next to first item(head)
    _currPtr->setNextFingerTable(_head);
    _currPtr  = NULL;


    printf("Finger Table Generation Completed\n\n");

    //search for specified finger table with nid
    int _nid = 0;

    //menu based finger table
    //prmopt user for node id and print the finger table for nodeid
    while(1){

        printf("Press ctrl + c to exit\n");
        printf("Finger Table For Node: ");
        scanf("%i",&_nid);
        printf("\n");

        _currPtr = _head;
        while(_currPtr != NULL){

            if(_currPtr->getNodeId() == _nid){
                _currPtr->displayFingerTable();
                FingerTable* _tempPtr = _currPtr->getNext();
                int _nextNodeId = _tempPtr->getNodeId();
                //printf("Next Node In Chain: %i\n\n",_nextNodeId);
                break;
            }
            else{
                _currPtr = _currPtr->getNext();
            }
        }

    }


    return 0;
}
```

OUPUT FOR NODE IDS = 377847 and 514068

```
Press ctrl + c to exit
Finger Table For Node: 377847

Node: 377847
---------------------
1    |  378010
2    |  378010
3    |  378010
4    |  378010
5    |  378010
6    |  378010
7    |  378010
8    |  378010
9    |  378200
10   |  378619
11   |  380175
12   |  380175
13   |  382156
14   |  386754
15   |  394314
16   |  411066
17   |  443802
18   |  508966
19   |  115781
---------------------

Press ctrl + c to exit
Finger Table For Node: 514068

Node: 514068
---------------------
1    |  514548
2    |  514548
3    |  514548
4    |  514548
5    |  514548
6    |  514548
7    |  514548
8    |  514548
9    |  514548
10   |  514861
11   |  515537
12   |  516225
13   |  518531
14   |  522361
15   |  6244
16   |  22602
17   |  56123
18   |  121039
19   |  252123
---------------------
```

Problem D

MakeFile

```
all:spread

spread: rumorSpreading.c
    gcc rumorSpreading.c -o spread

clean:
    rm -f spread
```

CODE

```
/*
 * @Author Swapnil Acharya
 * @since 4/4/2020
 */

#include <stdlib.h>
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <time.h>

/*
 * Struct to minit a node is epidemic protocol
 * Stuct constains follwing feilds
 * _nodeID: to store id of a node
 * _isInfected: boolean variable to indicated whtere the node is infected or not
 * _isRemoved: boolean variable to indicated where this node can spread infection or not
 */
typedef struct{
    int _nodeId;
    bool _isInfected;
    bool _isRemoved;
} Node;


#define MAX_NODES   1000000 //total nodes used for this simulaton
#define MAX_ROUNDS  1000 //total rounds useddd in this simulation


double PSTOP[5] = {0.2,0.4,0.6,0.8,1.00}; //probalbities where a node will stop spreading
infection

/*
 * This function returns the count of infected nodes in a given nodeList
 * @ param _nodeList List of Nodes
 * @ return count of susceptible nodes OR nodes that have not been infected
 */
int getNumberOfIgnorantNodes(Node*  _nodeList){
    int _count = 0;
    int i =0;
    for(i = 0; i < MAX_NODES; i++){
```

```
        //if the node is infected then increment the count
        if(  (*(_nodeList+ i))._isInfected == false ){
            _count = _count + 1;
        }
    }
    return _count; //return the count of infected nodes
}


/* This function initialized list of nodes
 * @param _nodeList List of nodes
 * @return true if the inifialization is sucessfull
 */
bool initializeNodes(Node*  _nodeList){
    int i = 0;
    for(i = 0; i < MAX_NODES;i++){
        (*(_nodeList+i))._nodeId = i; //give nodeid value of node index
        (*(_nodeList+i))._isInfected = false; //initialilly set node as susceptible
        (*(_nodeList+i))._isRemoved = false; //initially set node so that it can continue
to gossip
    }

    srand(time(0)); //seed random to be current time
    int _infectToIndex = rand() % MAX_NODES; //get a random index
    (*(_nodeList+_infectToIndex))._isInfected = true; //infect the randomly selected node

    return true;
}



int main(int _argc, char *_argv[]){

    //create Nodes by dynaically allocation
    Node* _nodeList = (Node *)malloc(MAX_NODES * sizeof(Node));

    //seed random
    srand(time(0));

    //start infection by rumor spreading
    printf("\n\nStarted Demonstration of Epidemic Protcol,\n by spreading disease via
Rumor Spreading\n\n");

    printf("Total Nodes: %i Total ROunds: %i\n\n",MAX_NODES,MAX_ROUNDS);

    int i = 0;
    int j = 0;
    int k = 0;
    float _prob = 0.00000;
    double _fractionIgnorant = 0.0000;
    int _ignorantNodes = 0;
    int _indexToInfect = 0;

    //times to find run time
    time_t _startTime, _endTime;
    time(&_startTime); //get current time

    for(k = 0; k < 5; k++){ //oop for PSTOP values

        initializeNodes(_nodeList); //intialize list of nodes

        for(i=0; i < MAX_ROUNDS; i++){ //do for maxmum number of rounds
```

```
            //EVERY NODE SELCTS RANDOM NODE TO EXCHANGE INFO
            for(j = 0; j < MAX_NODES; j++){

                if( (  (*(_nodeList+j))._isInfected == true) &&
( (*(_nodeList+j))._isRemoved == false ) ){

                    _indexToInfect = rand() % MAX_NODES; //select a random index for node
to infect

                    //if a radomlyy selected node is not infected, then infec the node
                    if( (*(_nodeList+_indexToInfect))._isInfected == false){
                        (*(_nodeList + _indexToInfect))._isInfected = true;
                    }
                    //else if a node is already infected then calculate probability that
it will stop spreading rumour
                    else{
                        _prob = (double) (rand() % MAX_NODES) / (double) MAX_NODES; //get
a probability betwwen 0 and 1
                        if(  _prob < PSTOP[k] ) {  //if the calcualted probility is less
thatn the given probability
                            (*(_nodeList + j))._isRemoved = true;  //then stop the node
from spreading rumors
                        }
                    }
                }

            }
        }

        _ignorantNodes = getNumberOfIgnorantNodes(_nodeList); //get number of suceptible
nodes
        _fractionIgnorant = (double)_ignorantNodes/(double)MAX_NODES; //get the faction
os suceptible nodes
        printf("PSTOP: %0.2f | S: %f |
IgnorantNodes: %i\n",PSTOP[k],_fractionIgnorant,_ignorantNodes); //display the results
    }

    //display elapsed time
    time(&_endTime);
    double _elapsedTime = difftime(_endTime,_startTime);
    printf("ElapedTIme: %0.3f seconds\n",_elapsedTime);

    //free memory allocated in heap for Node List
    free(_nodeList);

    printf("\nDesmostration of Epidemic Protocol  Complete\n\n");

    return 0;
}
```

OUPUT Part A

```
swapnil@Swapnil_Acharya:~$ ssh ea4963aw@stcloudstate@199.17.28.75
ea4963aw@stcloudstate@199.17.28.75's password:
Last login: Mon Apr  6 12:04:24 2020 from 97-88-13-138.dhcp.roch.mn.charter.com
[ea4963aw@ahscentos ~]$ cd CSCI_610
[ea4963aw@ahscentos CSCI_610]$ cd Homework2
[ea4963aw@ahscentos Homework2]$ cd code_problemD
[ea4963aw@ahscentos code_problemD]$ ls
Makefile  rumorSpreading.c  spread
[ea4963aw@ahscentos code_problemD]$ ./spread


Started Demonstration of Epidemic Protcol,
 by spreading disease via Rumor Spreading

Total Nodes: 1000000 Total ROunds: 1000

PSTOP: 0.20 | S: 0.002469 | IgnorantNodes: 2469
PSTOP: 0.40 | S: 0.034293 | IgnorantNodes: 34293
PSTOP: 0.60 | S: 0.087749 | IgnorantNodes: 87749
PSTOP: 0.80 | S: 0.145920 | IgnorantNodes: 145920
PSTOP: 1.00 | S: 0.203843 | IgnorantNodes: 203843
ElapedTIme: 51.000 seconds

Desmostration of Epidemic Protocol  Complete
```

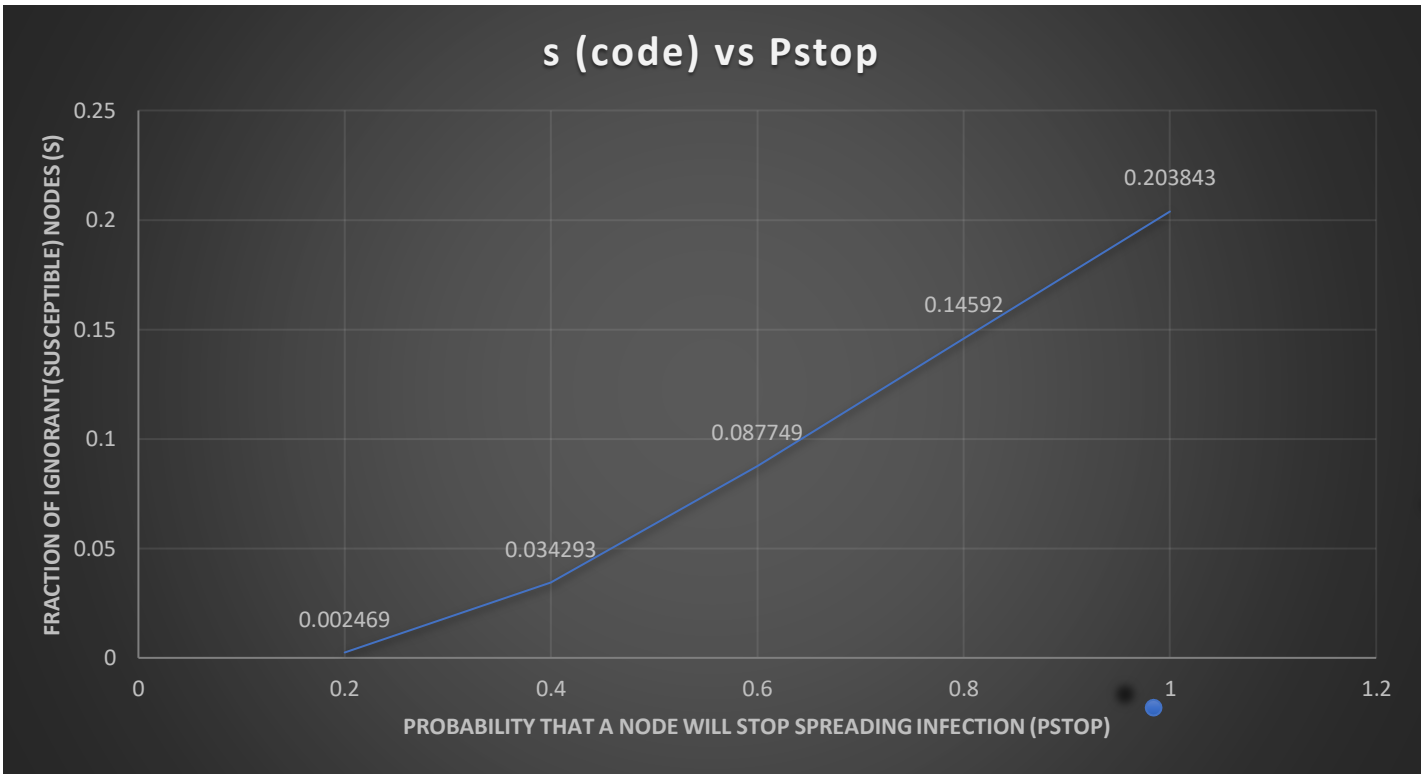
Output Part B

The fraction s was calculated using wolfram alpha online, please see pictures at the end of this document to see how these values were derived.

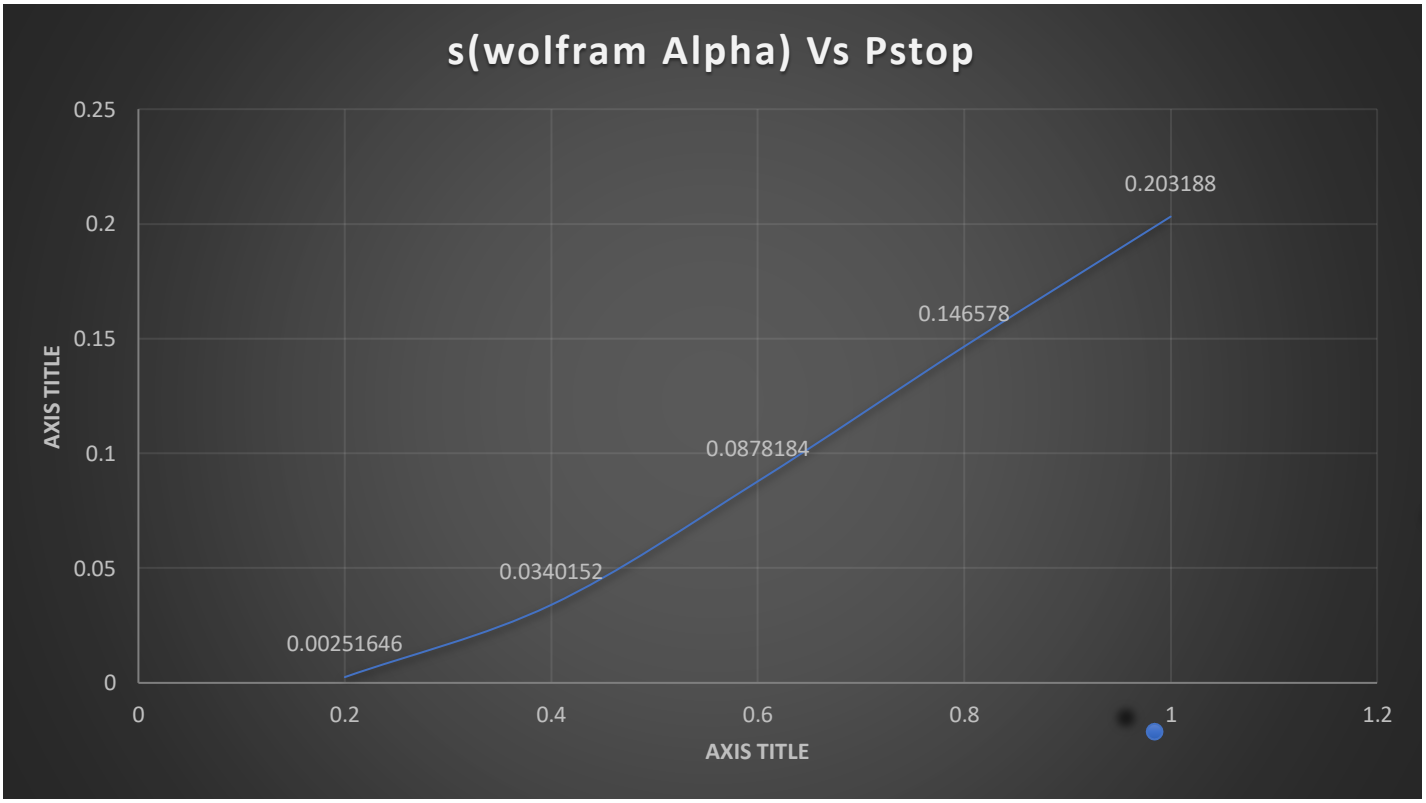
Pstop	s from wolfram Alpha
0.2	0.00251646
0.4	0.0340152
0.6	0.0878184
0.8	0.146578
1	0.203188

Output Part C

Part A Chart



Part B Chart



WOLFRAM ALPHA SOLITIONS:

PSTOP = 0.2

s = e ^ (-((1/0.2) + 1) (1-s))

Extended Keyboard | Upload | Examples | Random

Assuming "s" is a variable | Use as a unit instead

Assuming "s" is a variable | Use "1-s" as referring to math instead

Input:

$$s = e^{-(1/0.2+1)(1-s)}$$

Result:

$$s = e^{-6(1-s)}$$

Alternate forms:

$$s = 0.00247875\, e^{6s}$$

$$s = e^{6s-6}$$

Alternate form assuming s is real:

$$s = e^{6s-6} + 0$$

Number line:

Real solutions:

$$s \approx 0.00251646$$

$$s \approx 1$$

Solution: Approximate form

$$s \approx -0.166667\, W_n\left(-\frac{10\,811\,547}{726\,948\,227}\right), \quad n \in \mathbb{Z}$$

$W_k(z)$ is the analytic continuation of the product log function

$\mathbb{Z}$ is the set of integers

Integer solution:

$$s = 1$$

Download Page

POWERED BY THE **WOLFRAM LANGUAGE**

PSTOP = 0.4

s = e ^ (-((1/0.4) + 1) (1-s))

Extended Keyboard | Upload | Examples | Random

Assuming "s" is a variable | Use as a unit instead

Assuming "s" is a variable | Use "1-s" as referring to math instead

Input:

$$s = e^{-(1/0.4+1)(1-s)}$$

Result:

$$s = e^{-3.5\,(1-s)}$$

Alternate form:

$$s = 0.0301974\, e^{3.5\,s}$$

Number line:

Alternate form assuming s is real:

$$s = e^{3.5\,s-3.5} + 0$$

Real solutions:

$$s \approx 0.0340152$$

$$s \approx 1$$

Solution: Approximate form

$$s \approx -0.285714\, W_n\left(-\frac{52\,392\,277}{495\,712\,552}\right), \quad n \in \mathbb{Z}$$

$W_{k(z)}$ is the analytic continuation of the product log function

$\mathbb{Z}$ is the set of integers

Integer solution:

$$s = 1$$

Download Page

POWERED BY THE **WOLFRAM LANGUAGE**

PSTOP = 0.6

s = e ^ (-((1/0.6) + 1) (1-s))

Extended Keyboard | Upload | Examples | Random

Assuming "s" is a variable | Use as a unit instead

Assuming "s" is a variable | Use "1-s" as referring to math instead

Input:

$$s = e^{-(1/0.6+1)(1-s)}$$

Result:

$$s = e^{-2.66667\,(1-s)}$$

Alternate form:

$$s = 0.0694835\, e^{2.66667\,s}$$

Alternate form assuming s is real:

$$s = e^{2.66667\,s-2.66667} + 0$$

Number line:

Real solutions:

$$s \approx 0.0878184$$

$$s \approx 1.$$

Solution: Approximate form

$$s \approx -0.375\, W_n\left(-\frac{110532919}{596542686}\right), \quad n \in \mathbb{Z}$$

$W_k(z)$ is the analytic continuation of the product log function

$\mathbb{Z}$ is the set of integers

Integer solution:

$$s = 1$$

Download Page

POWERED BY THE **WOLFRAM LANGUAGE**

PSTOP = 0.8

s = e ^ (-((1/0.8) + 1) (1-s))

Extended Keyboard | Upload | Examples | Random

Assuming "s" is a variable | Use as a unit instead

Assuming "s" is a variable | Use "1-s" as referring to math instead

Input:

$$s = e^{-(1/0.8+1)(1-s)}$$

Result:

$$s = e^{-2.25\,(1-s)}$$

Alternate form:

$$s = 0.105399\, e^{2.25\,s}$$

Alternate form assuming s is real:

$$s = e^{2.25\,s-2.25} + 0$$

Number line:

Real solutions:

$$s \approx 0.146578$$

$$s \approx 1$$

Solution: Approximate form

$$s \approx -0.444444\, W_n\left(-\frac{18\,945\,075}{79\,887\,052}\right), \quad n \in \mathbb{Z}$$

$W_k(z)$ is the analytic continuation of the product log function

$\mathbb{Z}$ is the set of integers

Integer solution:

$$s = 1$$

Download Page

POWERED BY THE **WOLFRAM LANGUAGE**

P = 1.0

s = e ^ (-((1/1.0) + 1) (1-s))

Extended Keyboard | Upload | Examples | Random

Assuming "s" is a variable | Use as a unit instead

Assuming "s" is a variable | Use "1-s" as referring to math instead

Input:

$$s = e^{-(1/1+1)(1-s)}$$

Result:

$$s = e^{-2\,(1-s)}$$

Alternate forms:

$$s = 0.135335\, e^{2\,s}$$

$$s = e^{2\,s-2}$$

Alternate form assuming s is real:

$$s = e^{2\,s-2} + 0$$

Number line:

Real solutions:

$$s \approx 0.203188$$

$$s \approx 1$$

Solution: Approximate form

$$s \approx -0.5\, W_n\left(-\frac{55\,756\,553}{205\,994\,149}\right), \quad n \in \mathbb{Z}$$

$W_{k(z)}$ is the analytic continuation of the product log function

$\mathbb{Z}$ is the set of integers

Integer solution:

$$s = 1$$

Download Page

POWERED BY THE **WOLFRAM LANGUAGE**