Shashi Shivaraju
XID: C88650674

REPORT
ECE 6680 Lab #8 – Real Time Scheduling using RMA

Objective:

To perform Rate Monotonic Analysis (RMA) and determine if the given system design is schedulable.

System Design:

The system in consideration is **Intertial Navigation System (INS)**. It is a real-time shipboard avionic system which has strict time constraints for providing information to other shipboard devices. The top level design specifies that INS has to comply with the following timing constraints for its tasks:

Tasks	Period (ms)
Compute attitude data	2.56
Compute velocity data	40.96
Compute position data	1,280.00
Display data	1,000.00
Compose attitude message	61.44
Compose navigation message	1,024.00

The design specifies that the Tasks are periodic but not be independent. All tasks share the same result table (write mode for computational tasks, read mode for all others). The attitude and navigation message composition tasks share the same I/O channel. The system will run on a platform using a Motorola MC68302 microcontroller and a Linux real-time kernel, which offers a priority ceiling protocol. The overhead for this system is 153 µs per task. The estimated the execution times and resource usage times for each of the tasks is given below:

Task	Run time (ms)	Result table usage (ms)	I/O channel usage (ms)
attitude	1.30	0.20	-
velocity	4.70	0.20	-
position	3.00	0.20	-
display	23.00	0.30	-
att message	9.00	0.15	3.00
nav message	38.30	0.30	6.00

Shashi Shivaraju
XID: C88650674

Implementation:

Rate Monotonic Analysis for schedulability follows the steps listed below:

a) Assign priority to each task :

- Unique priorities to each task are set pre-execution. The shorter the period of the task the higher is its priority.

b) Calculate blocking time for priority ceiling protocol :

- For each task and the corresponding resources associated with it, maximum blocking time is calculated and total blocking time of the task is determined.

- In priority ceiling protocol, the maximum blocking time for a task (T_i) on resource (R_i) is the maximum of Direct Blocking and Push-through Blocking.

- Direct blocking occurs when higher priority task waits on the lower priority task and Push-through blocking occurs when medium priority tasks wait due to temporary priority inheritance.

- The total maximum blocking of each task (B_i) is the summation of maximum blocking time of the task (Ti) on each of the resources (Ri).

- Below table illustrates all the tasks along with their corresponding resources and execution details :

Task	Run time R (ms)	Period T (ms)	Priority (P_1 = Highest)	Resource 1: Result table usage (ms) R1 Ceiling Priority = P1				Resource 2: I/O usage (ms) R2 Ceiling Priority = P3				Total Max Blocking Bi (ms)
				Usage Time	Max. Block Direct	Max. Block Push Through	Max. Blocking for R1	Usage Time	Max. Block Direct	Max. Block Push Through	Max. Blocking for R2	
attitude	1.30	2.56	P1	0.20	0.30	0	**0.30**	0	0	0	**0**	**0.30**
velocity	4.70	40.96	P2	0.20	0.30	0	**0.30**	0	0	0	**0**	**0.30**
Att msg	9.00	61.44	P3	0.15	0.30	0	**0.30**	3.00	6.00	0	**6.00**	**6.30**
display	23.00	1000.00	P4	0.30	0.30	0	**0.30**	0	0	6.00	**6.00**	**6.30**
Nav msg	38.00	1024.00	P5	0.30	0.20	0	**0.20**	6.00	0	0	**0**	**0.20**
Position	3.00	1280.00	P6	0.20	0	0	**0**	0	0	0	**0**	**0**

Total Max Blocking Time Bi = Max. Blocking for R1 + Max. Blocking for R2

Shashi Shivaraju
XID: C88650674

c) Perform Rate Monotonic Analysis with overhead and blocking:

- Using the above illustrated table and below mentioned formula, schedulability test is performed:

[Formula:]

LHS

$$\min_{(k,l)} \sum_{j=1}^{i-1} [(R_j + O_v) * ceil(l * T_k / T_j)] + R_i + O_v + B_i <= (l * T_k)$$

RHS

Where,

n = number of tasks = 6;

∀i = 1 .... n;

Let k = 1 .... i;

Let l = 1 .... floor[T_i/T_k];

T_i = Period of Task i;

T_k = Period of task k;

O_v = System overhead = 153 micro seconds;

B_i = Total Max Blocking Time of Task i;

R_i = Runtime of Task i;

[Code Snippet]:

```
for(i = 1;i <= TOTAL_TASKS ; i++){ /* i = 1 .....n; n = no. of tasks*/
      isSchedulable = 0; /*Initialize to 0*/
      for(k = 1;k <= i; k++){ /* k = 1.....i */
            limit = (int)floor(Tasks[i-1].Period/Tasks[k-1].Period);
            for(l = 1;l <= limit;l++){ /* l = 1 .... |_(Ti/Tk)_| */
                  summation = 0; /* initialize to 0 */
                  condition = l * Tasks[k-1].Period; /* RHS = (l*Tk) */
                  for(j = 1;j < i;j++){
                        summation = summation + (Tasks[j-1].RunTime + SYSTEM_OVERHEAD)
                              *(float)ceil((l*Tasks[k-1].Period)/Tasks[j-1].Period);}
                  summation = summation +
                  Tasks[i-1].Max_BlockingTime + Tasks[i-1].RunTime + SYSTEM_OVERHEAD;
                  /*Schedulability Test Condition*/
                  if(summation <= condition){
                        isSchedulable = 1;
                        printf("Condition satisfied for
                        i = %d k = %d l = %d summation = %f condition = %f\n",\
                        i,k,l,summation,condition);
                        break; /*increment i*/
                        }
                        else
                        {
                        isSchedulable = 0;
                        printf("Condition NOT satisfied for i = %d k = %d l = %d
summation = %f condition = %f\n",i,k,l,summation,condition); /*increment l*/
                        }
                  }break; /*increments i*/
            }
}
```

Shashi Shivaraju
XID: C88650674

d) Rate Monotonic Analysis Result:

The Schedulability theorem results are summarized below:

Theorem Result	i	k	l [Single/Range]	LHS of Test	RHS of Test
Pass	**1**	**1**	**1**	**1.753000**	**2.560000**
Fail	2	1	1-4	**-**	**-**
Pass	**2**	**1**	**5**	**12.418000**	**12.800000**
Fail	3	1	1-22	-	-
Pass	**3**	**1**	**23**	**58.578000**	**58.880000**
Fail	4	1	1-85	-	-
Pass	**4**	**1**	**86**	**220.141000**	**220.160000**
Fail	5	1	1-157	-	-
Pass	**5**	**1**	**158**	**403.981000**	**404.480000**
Fail	6	1	1-164	-	-
Pass	**6**	**1**	**165**	**421.958000**	**422.400000**

- **[Conclusion]:** RMA determined that the designed system is **schedulable**.

Shashi Shivaraju
XID: C88650674

[Source Code]:

```
/*****************************************************************
*
*  FILE NAME      : header.h
*
*  DESCRIPTION  : Header file
*
*  PLATFORM       : Win32
*
*  DATE                  NAME                        REASON
*  16th April,2018       Shashi Shivaraju            ECE_6680_Lab_08
*                        [C88650674]
*****************************************************************/

/*Header file inclusions*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*Macro definations*/
#define SYSTEM_OVERHEAD 0.153 /*153 micro-seconds*/
#define TOTAL_TASKS           6

/*Structures definations*/
typedef struct __task
{
      int     priority;
      double RunTime;
      double Period;
      //float TableUsageTime;
      //float IOUsageTime;
      double Max_BlockingTime;
}task_data;


/* Function Prototypes */
int RateMonotonicAnalysis(task_data* Tasks);
```

```
/*****************************************************************
*
*  FILE NAME      : main.c
*
*  DESCRIPTION  : Implement Rate Monotonic Analysis for given set of tasks.
*
*  PLATFORM       : Win32
*
*  DATE                  NAME                        REASON
*  16th April,2018       Shashi Shivaraju            ECE_6680_Lab_08
*                        [C88650674]
*****************************************************************/
```

Shashi Shivaraju
XID: C88650674

```c
/* Header file inclusions */
#include "header.h"

int main(int argc, char *argv[])/* main function of the program */
{
	task_data* Tasks = NULL;			/* Pointer to memory */

	double RunTime[TOTAL_TASKS] = {1.30,4.70,9.00,23.00,38.30,3.00};
		/* Predefined run time of the tasks */
	double Period[TOTAL_TASKS] = {2.56,40.96,61.44,1000.00,1024.00,1280.00};
		/* Predefined period time of the tasks */
	double Max_BlockingTime[TOTAL_TASKS] = {0.30,0.30,6.30,6.30,0.20,0};
	 /* Calculated Max Blocking Time for eacch Tasks */
	int result = 0;				/* Variable to store return value */
	int i = 0;				/* Variable used in loop */

	/* Allocate memory to initialize Tasks */
	Tasks = (task_data*)calloc(TOTAL_TASKS,sizeof(task_data));
	if(!Tasks) /* error check */
	{
		printf("memory allocation failed"); /* display error message */
		return -1; /* return error code */
	}

	/*Initialize Tasks with the predefined data*/
	for(i = 0;i<TOTAL_TASKS;i++)
	{
		Tasks[i].priority = i+1;
		Tasks[i].RunTime = RunTime[i];
		Tasks[i].Period = Period[i];
		Tasks[i].Max_BlockingTime = Max_BlockingTime[i];
	}

	/* Perform Rate Monotonic Analysis to check if system is schedulable */
	result = RateMonotonicAnalysis(Tasks);
	if(0 == result)/*Display result of RMA*/
	{
		printf("System is schedulable !!!");
	}
	else
	{
		printf("System is NOT schedulable !!!");
	}

	/* Deallocate the Memory */
	if(Tasks)
	{
		free(Tasks);	/* free the memory */
		Tasks = NULL;
	}

	return 0; /* return success*/
}
```

Shashi Shivaraju
XID: C88650674

```
/*********************************************************************
*
*  FILE NAME      : functions.c
*
*  DESCRIPTION  : Modular functions used to implement
*                        Rate Monotonic Analysis.
*
*  PLATFORM       : Win32
*
*  DATE                  NAME                        REASON
*  16th April,2018       Shashi Shivaraju            ECE_6680_Lab_08
*                        [C88650674]
*********************************************************************/

/*Header file inclusions*/
#include "header.h"

int RateMonotonicAnalysis(task_data* Tasks)
{
      int isSchedulable = 0;                    /* Flag to represent Schedulability*/
      int result = 0,limit = 0;                 /* Variable to return value */
      int i = 0, l = 0,k = 0,j = 0;       /* Variables used in loop */
      double condition = 0;                     /* Variable used to store (l*Tk) */
      double summation = 0;                     /* Variable used to store LHS of
schedulability test condition */

      for(i = 1;i <= TOTAL_TASKS ; i++) /* i = 1 .....n; n = no. of tasks*/
      {
            isSchedulable = 0; /*Initialize to 0*/

            for(k = 1;k <= i; k++) /* k = 1.....i */
            {
                  limit = (int)floor(Tasks[i-1].Period/Tasks[k-1].Period);
                  for(l = 1;l <= limit;l++) /* l = 1 .... |_(Ti/Tk)_| */
                  {
                        summation = 0; /* initialize to 0 */
                        condition = l * Tasks[k-1].Period; /* RHS = (l*Tk) */

                        for(j = 1;j < i;j++)
                        {
                              summation = summation + (Tasks[j-1].RunTime +
SYSTEM_OVERHEAD)*(float)ceil((l*Tasks[k-1].Period)/Tasks[j-1].Period);
                        }

                        /*add max blocking time,runtime of the task and SYSTEM_OVERHEAD
to LHS*/
                        summation = summation + Tasks[i-1].Max_BlockingTime + Tasks[i-
1].RunTime + SYSTEM_OVERHEAD;

                        /*Schedulability Test Condition*/
                        if(summation <= condition)
                        {
                              isSchedulable = 1;
                              printf("Condition satisfied for i = %d k = %d l = %d
summation = %f condition = %f\n",\
                                    i,k,l,summation,condition);
                              break; /*increment i*/
                        }
                        else
                        {
```

Shashi Shivaraju
XID: C88650674

```
                                isSchedulable = 0;
                                printf("Condition NOT satisfied for i = %d k = %d l = %d
summation = %f condition = %f\n",i,k,l,summation,condition);
                                /*increment l*/
                          }
                    }
                    break; /*increments i*/
              }
        }

        if(isSchedulable)
              return 0;
        else
              return -1;
}
```

Shashi Shivaraju
XID: C88650674

[Result]:

Theorem Result	i	k	l	LHS of Test	RHS of Test
Pass	**1**	**1**	**1**	**1.753**	**2.56**
Fail	2	1	1	6.606	2.56
Fail	2	1	2	8.059	5.12
Fail	2	1	3	9.512	7.68
Fail	2	1	4	10.965	10.24
Pass	**2**	**1**	**5**	**12.418**	**12.8**
Fail	3	1	1	21.759	2.56
Fail	3	1	2	23.212	5.12
Fail	3	1	3	24.665	7.68
Fail	3	1	4	26.118	10.24
Fail	3	1	5	27.571	12.8
Fail	3	1	6	29.024	15.36
Fail	3	1	7	31.93	17.92
Fail	3	1	8	31.93	20.48
Fail	3	1	9	33.383	23.04
Fail	3	1	10	34.836	25.6
Fail	3	1	11	36.289	28.16
Fail	3	1	12	37.742	30.72
Fail	3	1	13	39.195	33.28
Fail	3	1	14	42.101	35.84
Fail	3	1	15	42.101	38.4
Fail	3	1	16	43.554	40.96
Fail	3	1	17	49.86	43.52
Fail	3	1	18	51.313	46.08
Fail	3	1	19	52.766	48.64
Fail	3	1	20	54.219	51.2
Fail	3	1	21	55.672	53.76
Fail	3	1	22	57.125	56.32
Pass	**3**	**1**	**23**	**58.578**	**58.88**
Fail	4	1	1	44.912	2.56
Fail	4	1	2	46.365	5.12
Fail	4	1	3	47.818	7.68
Fail	4	1	4	49.271	10.24
Fail	4	1	5	50.724	12.8
Fail	4	1	6	52.177	15.36
Fail	4	1	7	55.083	17.92
Fail	4	1	8	55.083	20.48
Fail	4	1	9	56.536	23.04
Fail	4	1	10	57.989	25.6
Fail	4	1	11	59.442	28.16
Fail	4	1	12	60.895	30.72
Fail	4	1	13	62.348	33.28

Fail	4	1	14	65.254	35.84
Fail	4	1	15	65.254	38.4
Fail	4	1	16	66.707	40.96
Fail	4	1	17	73.013	43.52
Fail	4	1	18	74.466	46.08
Fail	4	1	19	75.919	48.64
Fail	4	1	20	77.372	51.2
Fail	4	1	21	78.825	53.76
Fail	4	1	22	80.278	56.32
Fail	4	1	23	81.731	58.88
Fail	4	1	24	83.184	61.44
Fail	4	1	25	93.79	64
Fail	4	1	26	95.243	66.56
Fail	4	1	27	96.696	69.12
Fail	4	1	28	99.602	71.68
Fail	4	1	29	99.602	74.24
Fail	4	1	30	101.055	76.8
Fail	4	1	31	102.508	79.36
Fail	4	1	32	103.961	81.92
Fail	4	1	33	110.267	84.48
Fail	4	1	34	111.72	87.04
Fail	4	1	35	113.173	89.6
Fail	4	1	36	114.626	92.16
Fail	4	1	37	116.079	94.72
Fail	4	1	38	117.532	97.28
Fail	4	1	39	118.985	99.84
Fail	4	1	40	120.438	102.4
Fail	4	1	41	121.891	104.96
Fail	4	1	42	123.344	107.52
Fail	4	1	43	124.797	110.08
Fail	4	1	44	126.25	112.64
Fail	4	1	45	127.703	115.2
Fail	4	1	46	129.156	117.76
Fail	4	1	47	130.609	120.32
Fail	4	1	48	132.062	122.88
Fail	4	1	49	147.521	125.44
Fail	4	1	50	148.974	128
Fail	4	1	51	150.427	130.56
Fail	4	1	52	151.88	133.12
Fail	4	1	53	153.333	135.68
Fail	4	1	54	154.786	138.24
Fail	4	1	55	156.239	140.8
Fail	4	1	56	159.145	143.36
Fail	4	1	57	160.598	145.92

Fail	4	1	58	160.598	148.48
Fail	4	1	59	162.051	151.04
Fail	4	1	60	163.504	153.6
Fail	4	1	61	164.957	156.16
Fail	4	1	62	166.41	158.72
Fail	4	1	63	167.863	161.28
Fail	4	1	64	169.316	163.84
Fail	4	1	65	175.622	166.4
Fail	4	1	66	177.075	168.96
Fail	4	1	67	178.528	171.52
Fail	4	1	68	179.981	174.08
Fail	4	1	69	181.434	176.64
Fail	4	1	70	182.887	179.2
Fail	4	1	71	184.34	181.76
Fail	4	1	72	185.793	184.32
Fail	4	1	73	196.399	186.88
Fail	4	1	74	197.852	189.44
Fail	4	1	75	199.305	192
Fail	4	1	76	200.758	194.56
Fail	4	1	77	202.211	197.12
Fail	4	1	78	203.664	199.68
Fail	4	1	79	205.117	202.24
Fail	4	1	80	206.57	204.8
Fail	4	1	81	212.876	207.36
Fail	4	1	82	214.329	209.92
Fail	4	1	83	215.782	212.48
Fail	4	1	84	217.235	215.04
Fail	4	1	85	218.688	217.6
Pass	**4**	**1**	**86**	**220.141**	**220.16**
Fail	5	1	1	77.265	2.56
Fail	5	1	2	78.718	5.12
Fail	5	1	3	80.171	7.68
Fail	5	1	4	81.624	10.24
Fail	5	1	5	83.077	12.8
Fail	5	1	6	84.53	15.36
Fail	5	1	7	87.436	17.92
Fail	5	1	8	87.436	20.48
Fail	5	1	9	88.889	23.04
Fail	5	1	10	90.342	25.6
Fail	5	1	11	91.795	28.16
Fail	5	1	12	93.248	30.72
Fail	5	1	13	94.701	33.28
Fail	5	1	14	97.607	35.84
Fail	5	1	15	97.607	38.4

Fail	5	1	16	99.06	40.96
Fail	5	1	17	105.366	43.52
Fail	5	1	18	106.819	46.08
Fail	5	1	19	108.272	48.64
Fail	5	1	20	109.725	51.2
Fail	5	1	21	111.178	53.76
Fail	5	1	22	112.631	56.32
Fail	5	1	23	114.084	58.88
Fail	5	1	24	115.537	61.44
Fail	5	1	25	126.143	64
Fail	5	1	26	127.596	66.56
Fail	5	1	27	129.049	69.12
Fail	5	1	28	131.955	71.68
Fail	5	1	29	131.955	74.24
Fail	5	1	30	133.408	76.8
Fail	5	1	31	134.861	79.36
Fail	5	1	32	136.314	81.92
Fail	5	1	33	142.62	84.48
Fail	5	1	34	144.073	87.04
Fail	5	1	35	145.526	89.6
Fail	5	1	36	146.979	92.16
Fail	5	1	37	148.432	94.72
Fail	5	1	38	149.885	97.28
Fail	5	1	39	151.338	99.84
Fail	5	1	40	152.791	102.4
Fail	5	1	41	154.244	104.96
Fail	5	1	42	155.697	107.52
Fail	5	1	43	157.15	110.08
Fail	5	1	44	158.603	112.64
Fail	5	1	45	160.056	115.2
Fail	5	1	46	161.509	117.76
Fail	5	1	47	162.962	120.32
Fail	5	1	48	164.415	122.88
Fail	5	1	49	179.874	125.44
Fail	5	1	50	181.327	128
Fail	5	1	51	182.78	130.56
Fail	5	1	52	184.233	133.12
Fail	5	1	53	185.686	135.68
Fail	5	1	54	187.139	138.24
Fail	5	1	55	188.592	140.8
Fail	5	1	56	191.498	143.36
Fail	5	1	57	192.951	145.92
Fail	5	1	58	192.951	148.48
Fail	5	1	59	194.404	151.04

Fail	5	1	60	195.857	153.6
Fail	5	1	61	197.31	156.16
Fail	5	1	62	198.763	158.72
Fail	5	1	63	200.216	161.28
Fail	5	1	64	201.669	163.84
Fail	5	1	65	207.975	166.4
Fail	5	1	66	209.428	168.96
Fail	5	1	67	210.881	171.52
Fail	5	1	68	212.334	174.08
Fail	5	1	69	213.787	176.64
Fail	5	1	70	215.24	179.2
Fail	5	1	71	216.693	181.76
Fail	5	1	72	218.146	184.32
Fail	5	1	73	228.752	186.88
Fail	5	1	74	230.205	189.44
Fail	5	1	75	231.658	192
Fail	5	1	76	233.111	194.56
Fail	5	1	77	234.564	197.12
Fail	5	1	78	236.017	199.68
Fail	5	1	79	237.47	202.24
Fail	5	1	80	238.923	204.8
Fail	5	1	81	245.229	207.36
Fail	5	1	82	246.682	209.92
Fail	5	1	83	248.135	212.48
Fail	5	1	84	249.588	215.04
Fail	5	1	85	251.041	217.6
Fail	5	1	86	252.494	220.16
Fail	5	1	87	253.947	222.72
Fail	5	1	88	255.4	225.28
Fail	5	1	89	256.853	227.84
Fail	5	1	90	258.306	230.4
Fail	5	1	91	259.759	232.96
Fail	5	1	92	261.212	235.52
Fail	5	1	93	262.665	238.08
Fail	5	1	94	264.118	240.64
Fail	5	1	95	265.571	243.2
Fail	5	1	96	267.024	245.76
Fail	5	1	97	282.483	248.32
Fail	5	1	98	283.936	250.88
Fail	5	1	99	285.389	253.44
Fail	5	1	100	286.842	256
Fail	5	1	101	288.295	258.56
Fail	5	1	102	289.748	261.12
Fail	5	1	103	291.201	263.68

Fail	5	1	104	292.654	266.24
Fail	5	1	105	294.107	268.8
Fail	5	1	106	295.56	271.36
Fail	5	1	107	297.013	273.92
Fail	5	1	108	298.466	276.48
Fail	5	1	109	299.919	279.04
Fail	5	1	110	301.372	281.6
Fail	5	1	111	304.278	284.16
Fail	5	1	112	310.584	286.72
Fail	5	1	113	312.037	289.28
Fail	5	1	114	313.49	291.84
Fail	5	1	115	314.943	294.4
Fail	5	1	116	314.943	296.96
Fail	5	1	117	316.396	299.52
Fail	5	1	118	317.849	302.08
Fail	5	1	119	319.302	304.64
Fail	5	1	120	320.755	307.2
Fail	5	1	121	331.361	309.76
Fail	5	1	122	332.814	312.32
Fail	5	1	123	334.267	314.88
Fail	5	1	124	335.72	317.44
Fail	5	1	125	337.173	320
Fail	5	1	126	338.626	322.56
Fail	5	1	127	340.079	325.12
Fail	5	1	128	341.532	327.68
Fail	5	1	129	347.838	330.24
Fail	5	1	130	349.291	332.8
Fail	5	1	131	350.744	335.36
Fail	5	1	132	352.197	337.92
Fail	5	1	133	353.65	340.48
Fail	5	1	134	355.103	343.04
Fail	5	1	135	356.556	345.6
Fail	5	1	136	358.009	348.16
Fail	5	1	137	359.462	350.72
Fail	5	1	138	360.915	353.28
Fail	5	1	139	362.368	355.84
Fail	5	1	140	363.821	358.4
Fail	5	1	141	365.274	360.96
Fail	5	1	142	366.727	363.52
Fail	5	1	143	368.18	366.08
Fail	5	1	144	369.633	368.64
Fail	5	1	145	385.092	371.2
Fail	5	1	146	386.545	373.76
Fail	5	1	147	387.998	376.32

Fail	5	1	148	389.451	378.88
Fail	5	1	149	390.904	381.44
Fail	5	1	150	392.357	384
Fail	5	1	151	393.81	386.56
Fail	5	1	152	395.263	389.12
Fail	5	1	153	396.716	391.68
Fail	5	1	154	398.169	394.24
Fail	5	1	155	399.622	396.8
Fail	5	1	156	401.075	399.36
Fail	5	1	157	402.528	401.92
Pass	**5**	**1**	**158**	**403.981**	**404.48**
Fail	6	1	1	80.218	2.56
Fail	6	1	2	81.671	5.12
Fail	6	1	3	83.124	7.68
Fail	6	1	4	84.577	10.24
Fail	6	1	5	86.03	12.8
Fail	6	1	6	87.483	15.36
Fail	6	1	7	90.389	17.92
Fail	6	1	8	90.389	20.48
Fail	6	1	9	91.842	23.04
Fail	6	1	10	93.295	25.6
Fail	6	1	11	94.748	28.16
Fail	6	1	12	96.201	30.72
Fail	6	1	13	97.654	33.28
Fail	6	1	14	100.56	35.84
Fail	6	1	15	100.56	38.4
Fail	6	1	16	102.013	40.96
Fail	6	1	17	108.319	43.52
Fail	6	1	18	109.772	46.08
Fail	6	1	19	111.225	48.64
Fail	6	1	20	112.678	51.2
Fail	6	1	21	114.131	53.76
Fail	6	1	22	115.584	56.32
Fail	6	1	23	117.037	58.88
Fail	6	1	24	118.49	61.44
Fail	6	1	25	129.096	64
Fail	6	1	26	130.549	66.56
Fail	6	1	27	132.002	69.12
Fail	6	1	28	134.908	71.68
Fail	6	1	29	134.908	74.24
Fail	6	1	30	136.361	76.8
Fail	6	1	31	137.814	79.36
Fail	6	1	32	139.267	81.92
Fail	6	1	33	145.573	84.48

Fail	6	1	34	147.026	87.04
Fail	6	1	35	148.479	89.6
Fail	6	1	36	149.932	92.16
Fail	6	1	37	151.385	94.72
Fail	6	1	38	152.838	97.28
Fail	6	1	39	154.291	99.84
Fail	6	1	40	155.744	102.4
Fail	6	1	41	157.197	104.96
Fail	6	1	42	158.65	107.52
Fail	6	1	43	160.103	110.08
Fail	6	1	44	161.556	112.64
Fail	6	1	45	163.009	115.2
Fail	6	1	46	164.462	117.76
Fail	6	1	47	165.915	120.32
Fail	6	1	48	167.368	122.88
Fail	6	1	49	182.827	125.44
Fail	6	1	50	184.28	128
Fail	6	1	51	185.733	130.56
Fail	6	1	52	187.186	133.12
Fail	6	1	53	188.639	135.68
Fail	6	1	54	190.092	138.24
Fail	6	1	55	191.545	140.8
Fail	6	1	56	194.451	143.36
Fail	6	1	57	195.904	145.92
Fail	6	1	58	195.904	148.48
Fail	6	1	59	197.357	151.04
Fail	6	1	60	198.81	153.6
Fail	6	1	61	200.263	156.16
Fail	6	1	62	201.716	158.72
Fail	6	1	63	203.169	161.28
Fail	6	1	64	204.622	163.84
Fail	6	1	65	210.928	166.4
Fail	6	1	66	212.381	168.96
Fail	6	1	67	213.834	171.52
Fail	6	1	68	215.287	174.08
Fail	6	1	69	216.74	176.64
Fail	6	1	70	218.193	179.2
Fail	6	1	71	219.646	181.76
Fail	6	1	72	221.099	184.32
Fail	6	1	73	231.705	186.88
Fail	6	1	74	233.158	189.44
Fail	6	1	75	234.611	192
Fail	6	1	76	236.064	194.56
Fail	6	1	77	237.517	197.12

Fail	6	1	78	238.97	199.68
Fail	6	1	79	240.423	202.24
Fail	6	1	80	241.876	204.8
Fail	6	1	81	248.182	207.36
Fail	6	1	82	249.635	209.92
Fail	6	1	83	251.088	212.48
Fail	6	1	84	252.541	215.04
Fail	6	1	85	253.994	217.6
Fail	6	1	86	255.447	220.16
Fail	6	1	87	256.9	222.72
Fail	6	1	88	258.353	225.28
Fail	6	1	89	259.806	227.84
Fail	6	1	90	261.259	230.4
Fail	6	1	91	262.712	232.96
Fail	6	1	92	264.165	235.52
Fail	6	1	93	265.618	238.08
Fail	6	1	94	267.071	240.64
Fail	6	1	95	268.524	243.2
Fail	6	1	96	269.977	245.76
Fail	6	1	97	285.436	248.32
Fail	6	1	98	286.889	250.88
Fail	6	1	99	288.342	253.44
Fail	6	1	100	289.795	256
Fail	6	1	101	291.248	258.56
Fail	6	1	102	292.701	261.12
Fail	6	1	103	294.154	263.68
Fail	6	1	104	295.607	266.24
Fail	6	1	105	297.06	268.8
Fail	6	1	106	298.513	271.36
Fail	6	1	107	299.966	273.92
Fail	6	1	108	301.419	276.48
Fail	6	1	109	302.872	279.04
Fail	6	1	110	304.325	281.6
Fail	6	1	111	307.231	284.16
Fail	6	1	112	313.537	286.72
Fail	6	1	113	314.99	289.28
Fail	6	1	114	316.443	291.84
Fail	6	1	115	317.896	294.4
Fail	6	1	116	317.896	296.96
Fail	6	1	117	319.349	299.52
Fail	6	1	118	320.802	302.08
Fail	6	1	119	322.255	304.64
Fail	6	1	120	323.708	307.2
Fail	6	1	121	334.314	309.76

Fail	6	1	122	335.767	312.32
Fail	6	1	123	337.22	314.88
Fail	6	1	124	338.673	317.44
Fail	6	1	125	340.126	320
Fail	6	1	126	341.579	322.56
Fail	6	1	127	343.032	325.12
Fail	6	1	128	344.485	327.68
Fail	6	1	129	350.791	330.24
Fail	6	1	130	352.244	332.8
Fail	6	1	131	353.697	335.36
Fail	6	1	132	355.15	337.92
Fail	6	1	133	356.603	340.48
Fail	6	1	134	358.056	343.04
Fail	6	1	135	359.509	345.6
Fail	6	1	136	360.962	348.16
Fail	6	1	137	362.415	350.72
Fail	6	1	138	363.868	353.28
Fail	6	1	139	365.321	355.84
Fail	6	1	140	366.774	358.4
Fail	6	1	141	368.227	360.96
Fail	6	1	142	369.68	363.52
Fail	6	1	143	371.133	366.08
Fail	6	1	144	372.586	368.64
Fail	6	1	145	388.045	371.2
Fail	6	1	146	389.498	373.76
Fail	6	1	147	390.951	376.32
Fail	6	1	148	392.404	378.88
Fail	6	1	149	393.857	381.44
Fail	6	1	150	395.31	384
Fail	6	1	151	396.763	386.56
Fail	6	1	152	398.216	389.12
Fail	6	1	153	399.669	391.68
Fail	6	1	154	401.122	394.24
Fail	6	1	155	402.575	396.8
Fail	6	1	156	404.028	399.36
Fail	6	1	157	405.481	401.92
Fail	6	1	158	406.934	404.48
Fail	6	1	159	408.387	407.04
Fail	6	1	160	409.84	409.6
Fail	6	1	161	416.146	412.16
Fail	6	1	162	417.599	414.72
Fail	6	1	163	419.052	417.28
Fail	6	1	164	420.505	419.84
Pass	**6**	**1**	**165**	**421.958**	**422.4**