Joshua Silva
ECE468
Lab8

Task	Period (ms)	Run time (ms)	Result table usage (ms) (R1)	I/O channel usage (ms) (R2)	Disk usage (ms) (R3)
Compute attitude data (attitude)	10.56	1.3	0.2	-	2
Compute velocity data (velocity)	40.96	4.7	0.2	-	3
Compose attitude message (att message)	61.44	9	-	3	-
Display data (display)	100	23	0.3	-	-
Compose navigation message (nav message)	165	38.3	-	6	-
Run-time Built-In Test (BIT) (runtime BIT)	285	10	-	-	1
Compute position data (position)	350	3	0.2	-	3
Compose test message (test message)	700	2	-	2	-

Task	R1 Max Direct Blocking	R1 Max Push Through Blocking	R2 Max Direct Blocking	R2 Max Push Through Blocking	R3 Max Direct Blocking	R3 Max Push Through Blocking	Max Blocking R1	Max Blocking R2	Max Blocking R3	Max Blocking (ms)
Attitude	0.3	0	0	0	3	0	0.3	0	3	3.3
Velocity	0.3	0.3	0	0	3	3	0.3	0	3	3.3
ATT Message	0	0.3	6	0	0	3	0.3	6	3	9.3
Display	0.2	0.2	0	6	0	3	0.2	6	3	9.2
NAV Message	0	0.2	2	2	0	3	0.2	2	3	5.2
BIT	0	0.2	0	2	3	3	0.2	2	3	5.2
Position	0	0	0	2	0	0	0	2	0	2
TEST Message	0	0	0	0	0	0	0	0	0	0

Task	R (ms)	T (ms)	B (ms)
Attitude	1.3	10.56	3.3
Velocity	4.7	40.96	3.3
ATT Message	9	61.44	9.3
Display	23	100	9.2
NAV Message	38.3	165	5.2
BIT	10	285	5.2
Position	3	350	2
TEST Message	2	700	0

This system is schedulable

Works for the following values of (k, l)

- For i = 1: (k, l) = 1, 1
- For i = 2: (k, l) = 1, 1
- For i = 3: (k, l) = 1, 3
- For i = 4: (k, l) = 1, 7
- For i = 5: (k, l) = 2, 4
- For i = 6: (k, l) = 6, 1
- For i = 7: (k, l) = 4, 3
- For i = 8: (k, l) = 1, 43

Code:

```
/* rma.c
 * Rate Monotonic Analysis
 *
 * Purpose: To test the schedulability of a group of tasks using RMA
 *
 * Assumptions: User understands how RMA works. User will have to change arrays of task values to test different scenarios
 *
 * The program only uses an executable to run. Will run tasks from Lab8 assignment
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Begin RMA */
int main(int argc, char *argv[])
{
	/* Declare tasks, leave 0 in the array empty for simplicity in SUMMATION calculation */
	int i, j, k, l, num_L; //Handle incrementation in loops (Select each index in task arrays)
	int n = 8; //Number of tasks
	double run[9] = {0.00, 1.30, 4.70, 9.00, 23.00, 38.30, 10.00, 3.00, 2.00}; //Runtime in ms
	double period[9] = {0.00, 10.56, 40.96, 61.44, 100.00, 165.00, 285.00, 350.00, 700.00}; //Period in ms
	double block[9] = {0.00, 3.30, 3.30, 9.30, 9.20, 5.20, 5.20, 2.00, 0.00}; //Max Blocking in ms
	double overhead = 0.153; //Overhead = 153 us per task or 0.153 ms per task
	double sum, lTk; //Keep track of summation & l*Tk

	//Make sure user enters correct # of arguments
	if (argc != 1)
	{
		printf("Program use is ./rma\n");
		exit(0);
	}


	/* Begin formula for RMA. Go through each task */
```

```
	//For i = 1...n
	for (i = 1; i <= n; i++)
	{
		//For k = 1...i
		for(k = 1; k <= i; k++)
		{
			num_L = (int) floor(period[i]/period[k]);
			//printf("num_L is %d, floor is %lf\n", num_L, floor(period[i]/period[k]));

			//For l = 1...floor(Ti/Tk)
			for (l = 1; l <= num_L; l++)
			{
				sum = 0; //Reset sum for new run
				lTk = l*period[k];

				//Summation j= 1...i-1
				for (j = 1; j <= (i - 1); j++)
				{
					//Summation of (Rj + O)*ceiling(lTk/Tj)
					sum += (run[j] + overhead) * ceil(lTk / period[j]);
				}
				sum += run[i] + block[i]; //Summation is done, add in Ri + Bi
				//printf("Sum is %lf, lTk is %lf\n", sum, lTk);

				if(sum <= lTk) printf("(k, l) = %i, %i\n", k, l); //printf("i=%i (k=%i
l=%i) <= lTk = %lf.....SCHEDULABLE\n", i, k, l, lTk);
				//else printf("i=%i (k=%i l=%i) <= lTk = %lf.....NOT
SCHEDULABLE\n", i, k, l, lTk);
			}
		}
	}

	return 0;
}
```