ECE 198JL First Midterm Exam
Fall 2013

Tuesday, September 17th, 2013

Name: Austin McWilliams NetID: apmcwil2

Discussion Section:

10:00 AM	[ ] JD1	
11:00 AM	[ ] JD2	
12:00 PM	[ ] JD7	
1:00 PM	[ ] JD9	[ ] JDA
2:00 PM	[ ] JDB	
3:00 PM	[X] JDC	
4:00 PM	[ ] JD8	

- **Be sure your exam booklet has 9 pages.**
- **Be sure to write your name and lab section on each exam page.**
- **We have provided an ASCII table at the back.**
- **Do not tear the exam booklet apart; you can only detach the last page.**
- **Use backs of pages for scratch work if needed.**
- **This is a closed book exam. You may not use a calculator.**
- **You are allowed one handwritten 8.5 x 11" sheet of notes.**
- **Absolutely no interaction between students is allowed.**
- **Be sure to clearly indicate any assumptions that you make.**
- **The questions are not weighted equally. Budget your time accordingly.**
- **Don't panic, and good luck!**

Problem 1	15 points:
Problem 2	20 points:
Problem 3	15 points:
Problem 4	15 points:
Problem 5	20 points:
Problem 6	15 points:
Total	100 points:

Page: 2 Name: Austin McWilliams

14

Problem 1 (15 pts): Binary Representation

1. Convert the following decimal numbers to six-bit 2's complement binary representation. Show your work.

   22 11 5 2 1 0

   0 1 1 0 1 0

   a) 22_{10} = 010110

   -1

   32 16 8 4 2 1 0

   0 0 0 0 0 1 0 ✓

   b) -32_{10} = 100000 ✓

   100000 ⟷ 011111+1 = 100000

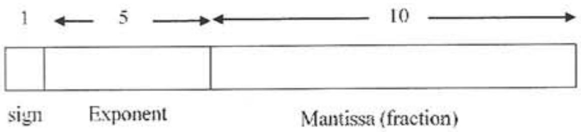
2. Convert the following eight-bit 2's complement binary number (given in hexadecimal notation) to decimal. Show your work.

   $BA = 10111010_2 \leftrightarrow 01000101+1 = 01000110$

   BA_{16} = -70 ✓

   $-(2^6+2^2+2^1) = -(64+4+2) = -70$

3. The IEEE 754 standard specifies *half-precision* binary floating-point format, called binary16, as follows:



The actual *normalized* number represented in this format is $(-1)^s \times 1.\text{fraction}_2 \times 2^{\text{exponent}-15}$.

   a) Convert the following decimal number to the 16-bit IEEE 754 *half-precision* floating point binary number representation. Show your work.

   ✓ ✓ ✓

   3.125 = 0100 0010 0100 0000

   Sign + → 0 $3_{10} = 11_2$ $.125_{10} = 0\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

   $3.125_{10} = 11.001_2 = 1.\underbrace{1001}_{\text{frac}} \times 2^1 \rightarrow 1+15 = 16$

   16 8 4 2 1

   0 0 0 0 1

   ✓ Exp: 10000

   b) Convert the following 16-bit IEEE 574 *half-precision* floating point binary number to the corresponding decimal representation. Show your work.

   1100110110000000 = -22 ✓

   ↑ Negative

   1.0110000000×2^4

   Exp: 10011

   $2^4+2^1+2^0 = 16+2+1 = 19$

   $19-15 = 4$

   10110.000000

   $2^4+2^2+2^1 = 22$ ✓

Page: 3 Name: Austin McWilliams

Problem 2 (20 pts): Operations on Binary Numbers

20 / 20

1. Perform the following arithmetic operations on eight-bit 2's complement numbers. Express your answer as an eight-bit 2's complement number in **hexadecimal** notation. Indicate if it has an *overflow* and/or *carryout of 1* by circling the corresponding YES or NO.

a) 01101101 - 10111011 = xB2 ✓ Overflow? [✓ YES] NO ✓ Carryout? YES [✓ NO] ✓

+ 01000101 01000100+1= 01000101

10110010

b) 10101101 + 10110110 = x63 ✓ Overflow? [✓ YES] NO ✓ Carryout? [✓ YES] NO ✓

+ 10110110

101100011

c) 10111111 - 10110100 = x0B ✓ Overflow? YES [✓ NO] ✓ Carryout? [✓ YES] NO ✓

+01001100 01001011+1=01001100

100001011

-128+32+16+8+4+2+1=-65 -128+32+16+4 -76

128

37

65

d) 01110101 + 00101110 = xA3 ✓ Overflow? [✓ YES] NO ✓ Carryout? YES [✓ NO] ✓

00101110

10100011

2. Add the following numbers in eight-bit 2's complement representation:

a) 00111100 + 011 = 00111111 ✓

00111100
+ 00000011
00111111

b) 00010001 + 10 = 00001111 ✓

00010001
+ 11111110
100001111

3. Recall Lab 2. Mathematically, 9000 + 0.0001 - 9005 = 9000 - 9005 + 0.0001. However, when left-side and right-side expressions are evaluated on the computer using 32-bit floating point representation, the results are:

9000 + 0.0001 - 9005 = -8998.099609 (xC60C9866)

9000 - 9005 + 0.0001 = -8997.099609 (xC60C9466)

Explain why this happens. ✓

These numbers are too large to be represented accurately by 32-bit floating point. For instance, when 0.0001 is added to 9000 and converted to floating point, the number of bits used in the fraction exceeds 23. Also, .0001 cannot be represented accurately in floating point because it cannot be evenly divided by any exponent of 2.

15

Problem 3 (15 pts): Codes, Error Detection and Correction

0000 0110 1010 1100

1. Code A has four code words {000, 011, 101, 110}. Would adding an even parity bit at the end of each code word allow us to detect 1 more bit of errors (circle one)?

   Answer: Yes [✓ No] $D_0 = 2$ Number of errors detected = 2-1 = 1
   $D_1 = 2$ " " = 2-1 = 1

2. Describe in three sentences or fewer how adding a parity bit can improve our ability to detect or correct errors in a transmission.

   Adding a parity bit can increase the distance of a code by up to an additional 1 unit. So if a code's distance were increased from 2-3, it could then detect 2 errors instead of 1 and correct 1 error instead of 0.

3. For 7-bit ASCII characters 'F' and 'G' write in binary their corresponding 8-bit **odd** parity representations. Place the parity bit in front of the 7-bit binary ASCII representation.

   'F' = 0 1000110 'G' = 1 1000111

   x46 = 0100 0110 x47 = 01000111

4. The hexadecimal number xBAAD44 can be interpreted as a string of 8-bit ASCII characters. An extra bit is placed at the beginning of each 7-bit ASCII character to make the 8-bit word.

   $xBA = 1011\ 1010_2$ $xAD = 10101101_2$ $x44 = 01000100$
   x3A x2D x44

   a) What is the extra bit for? (circle one) odd parity bit even parity bit [✓ not a parity bit]

   b) Write the characters encoded by the above ASCII string: :-D

5. Code B, consisting of four 5-bit code words, is a system critical code which needs to be able to **detect and correct 1-bit errors**. Suppose that 00000 is one legal code word. Identify a set of code words that would meet the specification for Code B.

Code Word 2	Code Word 3	Code Word 4
00111	11001	11110

Hint: What distance this code should have? 3

Problem 4 (15 pts): C program analysis

Consider the following "mystery" C program. Assuming that the user enters **6** when prompted to enter the number of terms, **trace** the execution of this program (you can make notes on this page or on the scratch pages if needed) to find the results of the computation performed. Answer the questions on the next page.

```
/* mystery.c */

#include <stdio.h>

int main()
{
   int n;
   int first = 0, second = 1;
   int next, c;

   printf("Enter the number of terms\n");
   scanf("%d", &n);

   printf("First %d terms are : ", n);

   for ( c = 0 ; c < n ; c++ )
   {
      if ( c <= 1 )
      {
         next = c;
      }
      else
      {
         next = first + second;
         first = second;
         second = next;
         /* CHECKPOINT FOR PART 1 */
      }
      printf("%d\n", next);
   }

   return 0;
}
```

Page: 6 Name: Austin McWilliams /2

1. At the location in the program marked CHECKPOINT FOR PART 1", determine and list the current values of the variables for each time that the program reaches that checkpoint. Fill in only as many rows as needed below.

-1	c =	0	first =	0	second =	1	next =	0
-1	c =	1	first =	0	second =	1	next =	1
	c =	2	first =	1	second =	1	next =	1
	c =	3	first =	1	second =	2	next =	2
	c =	4	first =	2	second =	3	next =	3
	c =	5	first =	3	second =	5	next =	5
	c =		first =		second =		next =	

2. Write down EXACTLY the formatted text that will be printed on the terminal screen by the program AFTER the user input has been provided.

First 6 terms are : 0 -/ [illegible] "\n"

0
1
1
2
3
5

3. Complete the following sentence to describe the exact computational task performed by this "mystery" program.

The "mystery.c" program computes and prints …. the Fibonacci sequence for the first n terms. Mathematically, it prints the values 0 and 1 and adds them, then adds the most recent addend (in this case 1) to the most recent sum (in this case 1) for the next "next" variable and repeats for n terms. ✓

20

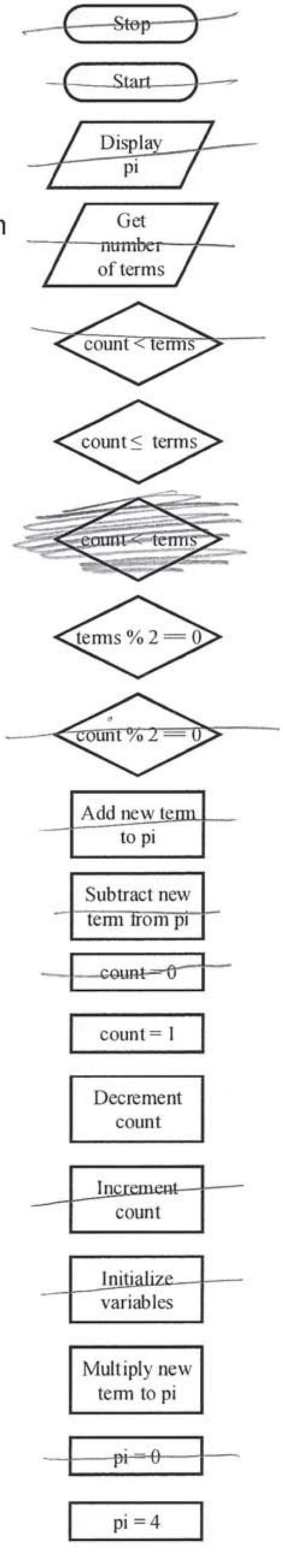
Problem 5 (20 pts): Algorithm design

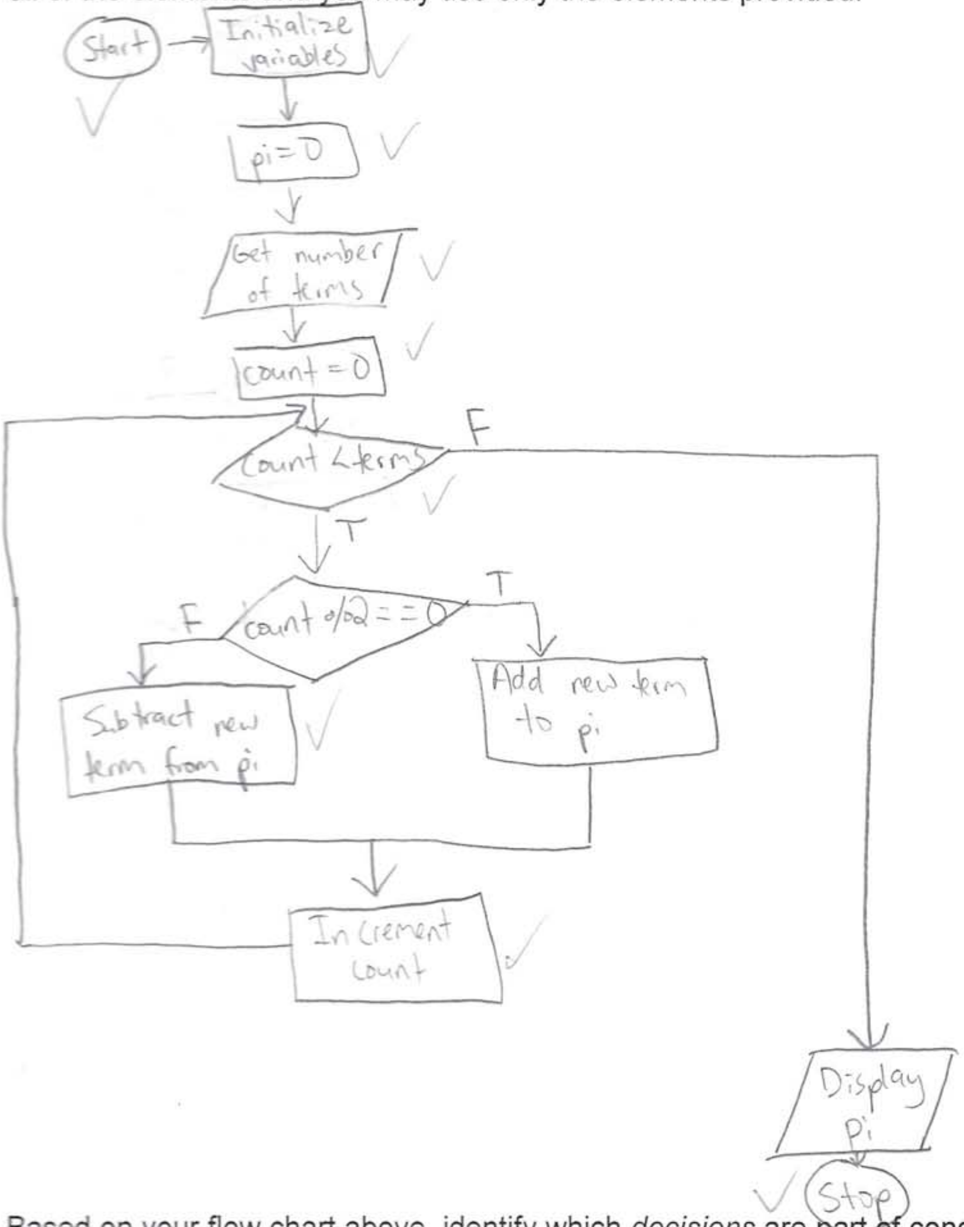
The value of π can be computed using the following series expansion:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \cdots + (-1)^n \frac{4}{2n+1} + \cdots$$

where $(-1)^n 4/(2n+1)$ is the expansion term.

1. Using the flow chart elements shown on the right, create a flow chart for a program that computes π for the number of terms entered by the user. For example, if the user enters 3, the program should compute $4 - 4/3 + 4/5$. You should not need all of the elements and you may use only the elements provided.





2. Based on your flow chart above, identify which *decisions* are part of conditional constructs and which are part of iterative constructs.

count < terms
iterative
for

count %2 == 0
conditional
if-else

Name: Austin McWilliams

15

Problem 6 (15 pts): Programming in C

Write in C a program that computes the value of π **according to your flowchart from Problem 5**. Program that exceeds 30 lines will not be graded. First line is already written for you.

Line #	Code
01	`#include <stdio.h>          /* Include standard input/output */`
02	`int main ()                 /* Enter the function */`
03	`{`
04	`int count, terms;           /*Intialize addends and user input */`
05	`float pi=0;                 /*Initialize final value of pi */`
06	
07	`printf("Number of terms used to calculate pi: \n"); /*Ask for input*/`
08	`scanf("%d", &terms);        /* Get user input */`
09	
10	`for (count=0; count<terms; count++)    /*Calculate pi */`
11	`{`
12	`if (count %2 ==0)`
13	`pi= pi + (4 /((2*count)+1));`
14	`else`
15	`pi= pi - (4 /((2*count)+1));`
16	`}`
17	
18	`printf ("pi =%f", pi);      /* Print pi */`
19	
20	`return 0;                   /* End function */`
21	`}`
22	
23	
24	
25	
26	
27	
28	
29	
30	

Page: 9 Name: Austin McWilliams

Table of ASCII Characters

Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex
(nul)	0	00	(sp)	32	20	@	64	40	`	96	60
(soh)	1	01	!	33	21	A	65	41	a	97	61
(stx)	2	02	"	34	22	B	66	42	b	98	62
(etx)	3	03	#	35	23	C	67	43	c	99	63
(eot)	4	04	$	36	24	D	68	44	d	100	64
(enq)	5	05	%	37	25	E	69	45	e	101	65
(ack)	6	06	&	38	26	F	70	46	f	102	66
(bel)	7	07	'	39	27	G	71	47	g	103	67
(bs)	8	08	(	40	28	H	72	48	h	104	68
(ht)	9	09	)	41	29	I	73	49	i	105	69
(nl)	10	0a	*	42	2a	J	74	4a	j	106	6a
(vt)	11	0b	+	43	2b	K	75	4b	k	107	6b
(np)	12	0c	,	44	2c	L	76	4c	l	108	6c
(cr)	13	0d	-	45	2d	M	77	4d	m	109	6d
(so)	14	0e	.	46	2e	N	78	4e	n	110	6e
(si)	15	0f	/	47	2f	O	79	4f	o	111	6f
(dle)	16	10	0	48	30	P	80	50	p	112	70
(dc1)	17	11	1	49	31	Q	81	51	q	113	71
(dc2)	18	12	2	50	32	R	82	52	r	114	72
(dc3)	19	13	3	51	33	S	83	53	s	115	73
(dc4)	20	14	4	52	34	T	84	54	t	116	74
(nak)	21	15	5	53	35	U	85	55	u	117	75
(syn)	22	16	6	54	36	V	86	56	v	118	76
(etb)	23	17	7	55	37	W	87	57	w	119	77
(can)	24	18	8	56	38	X	88	58	x	120	78
(em)	25	19	9	57	39	Y	89	59	y	121	79
(sub)	26	1a	:	58	3a	Z	90	5a	z	122	7a
(esc)	27	1b	;	59	3b	[	91	5b	{	123	7b
(fs)	28	1c	<	60	3c	\	92	5c	|	124	7c
(gs)	29	1d	=	61	3d	]	93	5d	}	125	7d
(rs)	30	1e	>	62	3e	^	94	5e	~	126	7e
(us)	31	1f	?	63	3f	_	95	5f	(del)	127	7f