ECEN 5013: Embedded Software Essentials

Homework 1: Embedded, Numbers & C-Programming Review

Question 1:

a) Hex: 0XDEADBEEF Given

Binary - 1101 1110 1010 1101 1011 1110 1110 1111

Decimal: (from Hex) $(15\times16^0) + (14\times16^1) + (16\times256) + (11\times16^3) + (3\times16^4) + (10\times16^5) + (14\times16^6) + (13\times16^7)$

$= 15 + 224 + 3584 + 45056 + 8,51,968 + 1,04,85,760 + 23,48,81,026 + 3,48,96,60,928$

$= 3,73,59,28,559$

Octal from Binary = 011 011 110 101 011 011 011 111 011 101 111

$\Rightarrow$ 3 3,6 5 3,3 3 7,3 5 7

Data type - uint32_t

b) Given: Decimal: 310,422

2	310,422	0
2	155211	1
2	77605	1
2	38802	0
2	19401	1
2	9700	0
2	4850	0
2	2425	1
2	1212	0
2	606	0
2	303	1
2	151	1
2	75	1
2	37	1
2	18	0
2	9	1
2	4	0
2	2	0
	1	1

Binary: 0 1 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 1 1 0

Hex: 4 8 C 9 6 i.e 0X00048C96

Octal: 001 001 011 110 010 010 110

$\Rightarrow$ 1 1 3 6 2 2 6

Data type: uint32_t, int32_t

c) 1,465 - Decimal

Binary:

2	1465	1
2	732	0
2	366	0
2	183	1
2	91	1
2	45	1
2	22	0
2	11	1
2	5	1
2	2	0
	1	1

Binary = 0101 1011 1001

Hex = 5 B 9 i.e 0x5B9

Octal: 010 110 111 001

: 2671

Data Type => uint16-t, int16-t

d) Given: 0b 1110 1010 1110

Decimal: $(0\times1) + (1\times2) + (1\times4) + (1\times8) + (0\times2^4) + (1\times2^5) + (0\times2^6) + (1\times2^7) + (0\times2^8) + (1\times2^9) + (1\times2^{10}) + (1\times2^{11})$

$= 0 + 2 + 4 + 8 + 0 + 32 + 0 + 128 + 0 + 512 + 1024 + 2048$

$= 3758$

Hex: 1110 1010 1110

⇒ 0xEAE i.e 0x0EAE

Octal: 111 010 101 110

⇒ 7256

Data type: uint16-t, int16-t

e) Given: 0x8FA44

decimal: $(4\times16^0) + (4\times16^1) + (10\times16^2) + (15\times16^3) + (8\times16^4)$

$= 588,356$

	8	F	A	4	4
Binary from hex ⇒	1000	1111	1010	0100	0100

Octal from Bin ⇒ 010 001 111 101 001 000 100

⇒ 2175104

Data type: uint32-t, int32-t

f) Given decimal: -172
datatype - int16-t

Binary:

2	172	0
2	86	0
2	43	1
2	21	1
2	10	0
2	5	1
2	2	0
	1	1

⇒ 0000 0000 1010 1100

1st compliment: 1111 1111 0101 0011

2s compliment: 1111 1111 0101 0100

Hex: 0xFF54

Octal: 001 111 111 101 010 100

⇒ 177524

g) Given decimal: 172

datatype: uint16-t, int16-t, uint8-t

Binary:

2	172	0
2	86	0
2	43	1
2	21	1
2	10	0
2	5	1
2	2	0
	1	1

⇒ 0000 0000 1010 1100

Hex: 0xAC

Octal: 010 101 100

⇒ 254

Table:

Decimal	Binary	~~Hexadecimal~~ Hexadecimal	Datatype
3,735,928,559	0b1101 1110 1010 1101 1011 1110 1110 1111	0xDEADBEEF	uint32-t
310,422	0b0100 1011 1100 1001 0110	0x48C96	uint32-t, int32-t
1,465	0b0 101 1011 1001	0x5B9	uint16-t, int16-t
3758	0b 1110 1010 1110	0xEAE	uint16-t, int16-t
588,356	0b 1000 1111 1010 0100 0100	0x8FA44	uint32-t, int32-t
-172	0b 1111 1111 0101 0100	0xFF54	int16-t
172	0b 1010 1100	0xAC	uint16-t, int16-t, uint8-t

Question 2:

- 124.84375

2 | 124	0
2 | 62	0
2 | 31	1
2 | 15	1
2 | 7	1
2 | 3	1
1	1

01111100

0.84375 × 2 = 1.68750 → 1
0.68750 × 2 = 1.37500 → 1
0.37500 × 2 = 0.75000 → 0
0.75 × 2 = 1.50 → 1
0.50 × 2 = 1.0 → 1

⇒ 11011

fixed point: 01111100.11011000

32 bit precision:

1.11110011011000×2^6 exp: 6+127 = 133 ⇒ 10000101

2 | 133	1
2 | 66	0
2 | 33	1
2 | 16	0
2 | 8	0
2 | 4	0
2 | 2	0
1	1

Sign bit | exponent | mantissa

0 - 10000101 - 11110011011000000000000

- -12.45897

2 | 12	0
2 | 6	0
2 | 3	1
1	1

⇒ 00001100

2's complement ⇒

00001100
11110011 + 1
11110100

0.45897 × 2 = 0.91794 → 0
0.91794 × 2 = 1.83588 → 1
0.83588 × 2 = 1.67176 → 1
0.67176 × 2 = 1.34352 → 1
0.34352 × 2 = 0.68704 → 0
0.68704 × 2 = 1.37408 → 1
0.37408 × 2 = 0.74816 → 0
0.74816 × 2 = 1.49632 → 1

⇒ 0.1110101

fixed point:
⇒ 11110100.01110101

32 bit precision: ~~1.11101000111010101×2^7~~

→ To be continued

Question 2

- -12.45897

32 bit floating :

12 in binary 0000 1100

0.45897 binary → .01110101

12.45897 = 1100.01110101

= 1.10001110101×2^3

sign bit | exp | Mantisa

1 - 1000 0010 - 10001110101111111111111

exp = 3 + 127 = 130

2	130	0
2	65	1
2	32	0
2	16	0
2	8	0
2	4	0
2	2	0
	1	

Question 3

- 0XC141-FEC0

1100 0100 0100 0001 1111 1110 1100 0000

sign bit = 1

Mantisa 1. 100 0001 1111 1110 1100

exponent 10001000 ⇒ $2^7 + 2^3$

⇒ 128 + 8 = 136

= 136 - 127 = 9

11000.001111111101100

⇒ 11000.001111111101100

$2^9 + 2^8 + 4 + 2 + 1$

= 512 + 256 + 4 + 2 + 1

= 775

fraction: $(1\times2^{-1}) + (1\times2^{-2}) + (1\times2^{-5}) + (1\times2^{-4}) + (1\times2^{-5}) + (0\times2^{-6}) + (1\times2^{-7}) + (1\times2^{-8}) + (0\times2^{-9}) + (0\times2^{-10})$

⇒ 0.5 + 0.25 + 0.125 + 0.0625 + 0.0313 + 0.0078 + 0.0039

⇒ 0.9805

⇒ -775.9805

=

X wrong calculation

⇒ intigral: $(1\times2^4) + (1\times2^3) = 24$

fraction: $(0\times2^{-1}) + (0\times2^{-2}) + (0\times2^{-3}) + (1\times2^{-4}) + (1\times2^{-5}) + (1\times2^{-6}) + (1\times2^{-7}) + (1\times2^{-8}) + (1\times2^{-9}) + (1\times2^{-10}) + (0\times2^{-11}) + (1\times2^{-12}) + (1\times2^{-13})$

= 0.1250 + 0.0625 + 0.0313 + 0.0156 + 0.0078 + 0.0039 + 0.0020 + 0.0010 + 0.0002 + 0.0001

= 0.2494

Question 3

- 0x46FE-8000

0100 0110 1111 1110 1000 0000 0000 0000

sign bit | exponent | Mantissa

0 | 10001101 | 11111101000000000000

exponent = $2^7 + 2^3 + 2^2 + 2^0$

$\Rightarrow 128+8+4+1$

$= 141$

$\Rightarrow 141 - 127 = 14$

$\therefore 1.1111101000000000000 \times 2^{14}$

$\Rightarrow 11111101000000.0000000$

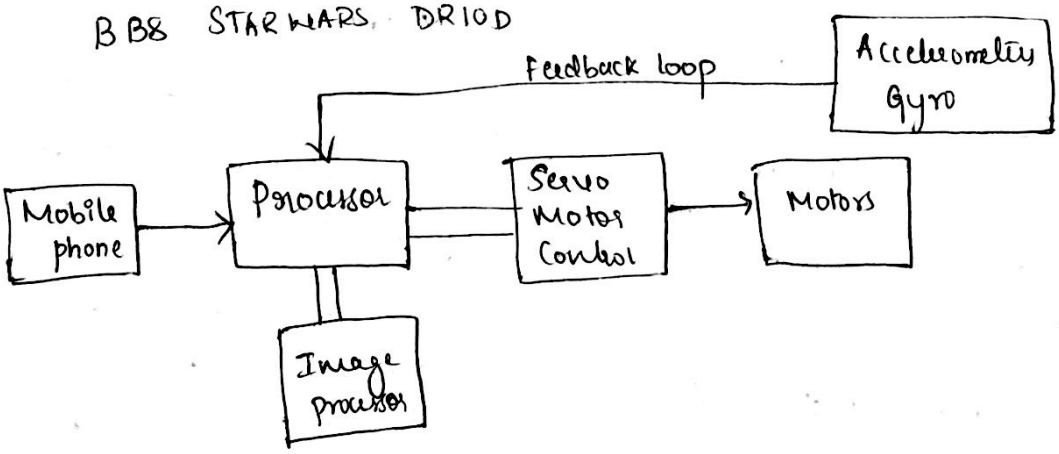
$\Rightarrow 2^{14} + 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^6$

$= 16384 + 8192 + 4096 + 2048 + 1024 + 512 + 256 + 64$

$= 32,576.0$

Question 4:

BB8 STAR WARS DROID



Question 5:

At 100, Data = 0xFE

After

0xF1 & 127

⇒ 11110001
01111111
01110001

⇒ 0x71

127 to Binary

⇒

2	127	1
2	63	1
2	31	1
2	15	1
2	7	1
2	3	1
	1	

⇒ 01111111

At 101 ⇒ Data = 0x34

0x34 = 00110100 in Binary

0x34 + 17 (decimal)

17 ⇒ 00010001 in Binary

So, 00110100
00010001
01000101

⇒ 0x45

At 103 ⇒ 15%4

⇒ 4|15
3

Remainder = 3

∴ ⇒ 0x03

At 102 ⇒ Data = 0x8C

⇒ ~~1000 1101 >>4~~ 0100 1101 >>4

⇒ ~~00000~~ 00000100

⇒ 0x08

At 104 ⇒ Data = 0x61

After ⇒ 0x61

At 105 ⇒ Data = 0x28

After : 1<<5 | 4<<2

1<<5 ⇒ 00000001

So, 00100000

4<<2 ⇒ 00000100

So, 00010000

At 105,

```
  0010 0000
  0001 0000
-------------
  0011 0000
    3    0
```

⇒ 0X30

At 106 ⇒ Data = 0X23

After = 0X23

At 107 ⇒ Data = 0X40

After = 22

in binary → 0001 0110
1 6

⇒ 0X16

Addres	Data	Data After
100	0XFE	0X71
101	0X34	0X45
102	0X8C	0X08
103	0X40	0X03
104	0X61	0X61
105	0X26	0X30
106	0X23	0X23
107	0X40	0X16

Question 6:

```
#include<stdio.h>

int main()
{
unsigned char arr [8];
int index=0;
arr[0] = 0xFE;
unsigned char * ptr = arr;
printf("Addr: %d\t%x\t", index, arr[index]);

arr[0]= 0xF1 & 127;

printf("%x\n", arr[index]);

index++;

arr[1]=0x34;
printf("Addr: %d\t%x\t", index, arr[index]);

arr[1]+=17;
printf("%x\n", arr[index]);

index+=2;

arr[3]=0x61;
printf("Addr: %d\t%x\t", index, arr[index]);

arr[3]=15%4;
printf("%x\n", arr[index]);

index--;
arr[2]=0x8C;
printf("Addr: %d\t%x\t", index, arr[index]);
arr[2]>>=4;
printf("%x\n", arr[index]);

index=4;
arr[4]=0x61;
printf("Addr: %d\t%x\t", index, arr[index]);
printf("%x\n", arr[index]);


index=5;
arr[5]=0x28;
printf("Addr: %d\t%x\t", index, arr[index]);
```

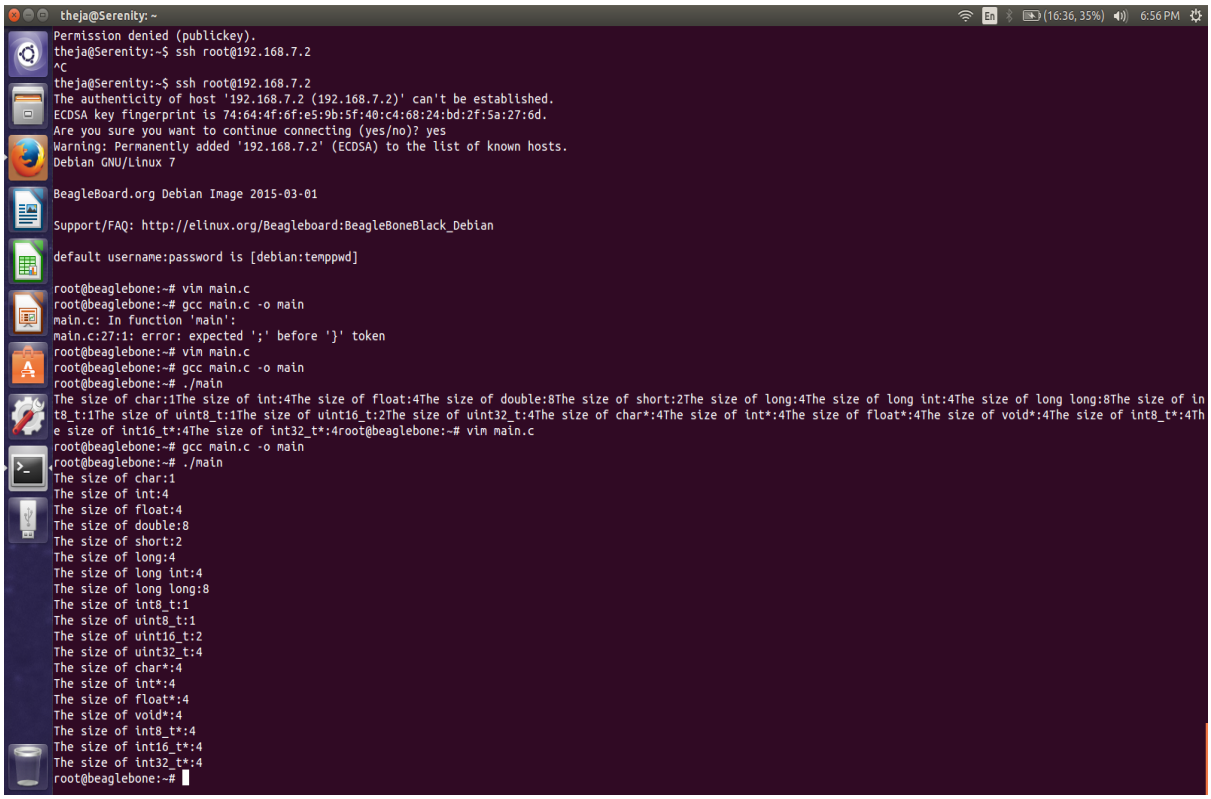
```
arr[5]= (1<<5)|(4<<2);
printf("%x\n", arr[index]);

index=6;
arr[6]=0x23;
printf("Addr: %d\t%x\t", index, arr[index]);
printf("%x\n", arr[index]);

index=7;
arr[7]=0x40;
printf("Addr: %d\t%x\t", index, arr[index]);
arr[7]=22;
printf("%x\n", arr[index]);

return 0;
}
```

Output:

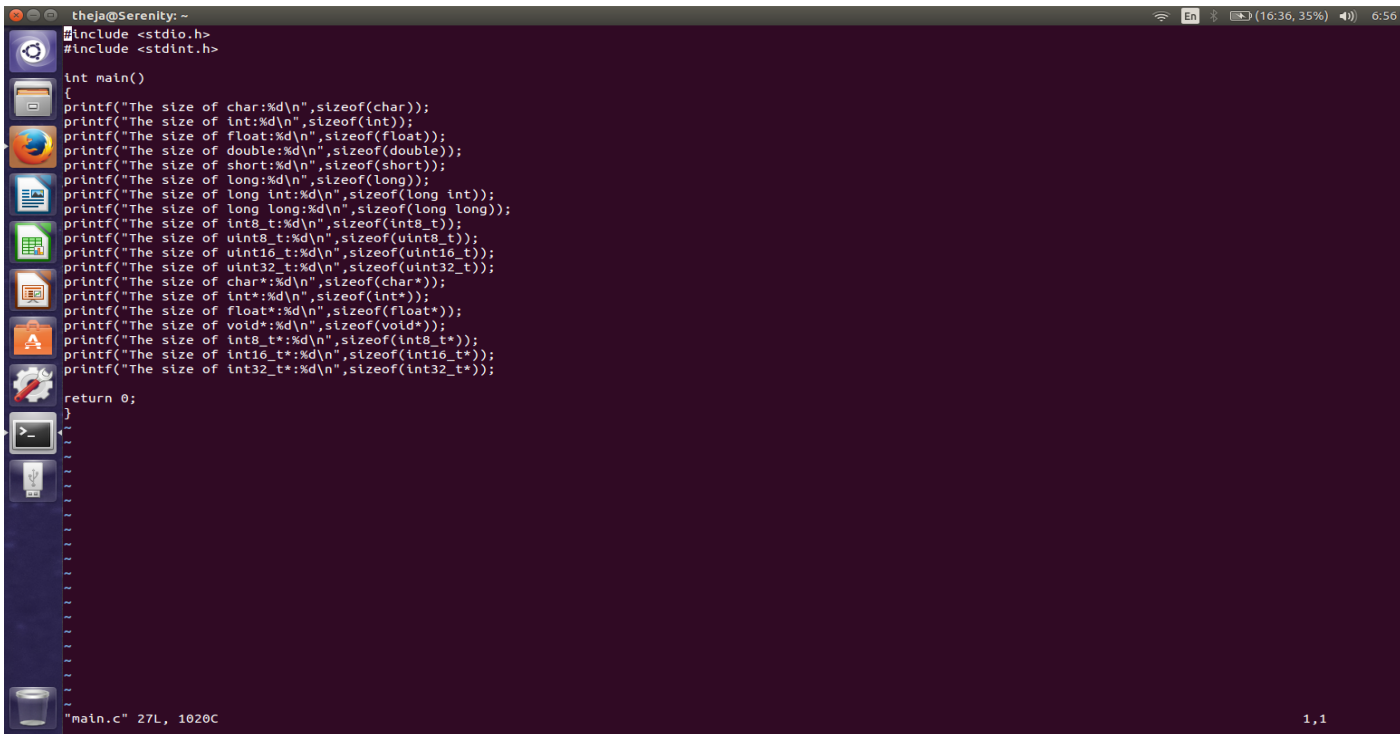
```
Addr: 0   fe   71
Addr: 1   34   45
Addr: 3   61   3
Addr: 2   8c   8
Addr: 4   61   61
Addr: 5   28   30
Addr: 6   23   23
Addr: 7   40   16
```

Question 7:

OUTPUT:



CODE:



Question 8: Pseudo code

If the length is negative

Print “enter a positive integer”

If the length is a character

Print ”enter a valid integer”

If the length is less than or greater than the string length

Print “enter the appropriate length of the string”

If the string contains only the null character

Print “enter a valid string”

If the string contains only white spaces

Print “enter a valid string”

If the string contains letters, digits , special characters or combination of all of them

Print “reverse operation successful “

Question 9:

```
#include <stdio.h>
void reverse(char *str, int length);
int main(void) {
   char word[]="abc";
   reverse(word,3);
}

void reverse(char *str, int len)
{
   char *begin, *last, *k, temp, temp2;
   int l=0;
if(len<=0)
   {
          printf("enter a positive integer for length");
          return 1;
}
```

```
if(*str="\0")
        {
                printf("enter a valid string");
                return 1;
        }

   begin=str;
   last=str+len-1;
   k=str;

   while(begin<last)
   {
      temp=*begin;
      *begin=*last;
      *last=temp;
      last--;
      begin++;
      l++;
      }
      while(*k!='\0')
      {
         printf("%c", *k);
         k++;
      }
return 0;
}
```