Northwestern University

To: Dr. Marchuk

From: Allen Liu

Subject: ME333 Mechatronics Winter Break Assignment

Due Date: 4^{th} Jan, 2024

Ex. A.1

As shown in HelloWorld.c and Makefile file. To compile and run the code simply run

```
make all
./HelloWorld.exe
```

To clean the build and executable files, run

```
make clean
```

Ex. A.2

The pointer variable is the variable that contains the memory address of another variable. It is different from the non-pointer variable since it does not stores the data directly, in order to access the value of the pointer-variable, it is required to go to the address in memory and get the value from there.

Ex. A.3

The interpreted code means the code are converted to the machine code, the executable code, while the program is running. The compiled code means the code is converted to the machine code before it is run.

Ex. A.4

- 0x1E = 0b0001 1110
- 0x32 = 0b0011 0010
- 0xFE = 0b1111 1110

Ex. A.6

For each byte, there are 2^8 bits, so there are total $2^{16} \times 2^8 = 2^{24}$ bits in the memory.

Ex. A.7

- 'k' = 107
- '5' = 53

- '-' = 45
- '?' = 63

Ex. A.8

- unsigned char: 0 to 255
- short: -32768 to 32767
- double: 2.3×10^{-308} to 1.7×10^{308}

Ex. A.10

The most significant bit of the signed integer is the sign bit. The 1 denotes that it is negative, while the 0 denotes that it is positive.

Ex. A.11

*int*s vs. *char*s

- **PROS**: The *char*s uses less bit than *int*s so that when performing the integer math, *char*s uses less resources than *int*s. So that it has better performance than *int*s when doing the integer math.
- **CONS**: Since the *char*s have less bits so that it can perform the operation in less range of values. Thus it is less reliable than *int*s.

*float*s vs. *double*s

- **PROS**: Similar to the previous one, the *float*s uses less bits than the *double*s. So than when performing mathematical operations, the *float*s utilize less resource than the *double*s, thus it is more efficient.
- **CONS**: Since the *float*s use less bits than *double*s, so that it is more likely to overflow when performing calculation on large values. And since *double*s use more bits, so that it is more precise than the *float*s. Hence the *float*s are less reliable than *double*s.

*char*s vs. *float*s

- **PROS**: The chars are less computationally expensive than floats. Since it requires more resources to performing floating point operations than integers, so that it is more effective.
- **CONS**: The floats are more precise than chars, so that the floats are more reliable than chars.

Ex. A.16

Since the pointer is the address of a variable in memory, so it can be represented as an integer with the same number of bytes. Since it is a 64-bit system, so that the address is a 8-byte value, so that it can be represented as a *unsigned long*, which is also a 8-byte value.

Ex. A.17

variable	i	j	kp	np	instruction
address	0xb0 - 0xb3	0xb4 - 0xb7	0xb8	0xb9	
a	unknown	unknown	unknown	unknown	
b	unknown	unknown	0xb0	unknown	kp = &i;
c	unknown	unknown	0xb0	unknown	j = *kp;
d	0xae	unknown	0xb0	unknown	i = 0xAE;
e	0xae	unknown	0xb0	0xb0	np = kp;
f	0x12	unknown	0xb0	0xb0	*np = 0x12;
g	0x12	0x12	0xb0	0xb0	j = *kp;

Table 1: Value of each variable after certain instruction