Quiz #1*

[CS 120] High-level Programming I: The C Programming Language

DigiPen INSTITUTE OF TECHNOLOGY SINGAPORE

GOH WEI ZHE	weizhe.goh	240919	54 /100
Name	Student login	Date	Points

Notes: This is a short-answer, closed books quiz. Do not collaborate or copy other people's work. Please write legibly – I can give points only for correct, clear answers I am able to read.

1. Building an executable

Building an executable from the source code in the C language compilers is a 4-stage process. In the table below write down a name of each stage, identify its product (a type of an output file), and describe the purpose of including such a stage. The stages must be listed in the order of execution:
(3 marks for each cell; 36 marks total)

#	Stage name	Product	Description
1	Preprocessor	.s	It will ~~read~~ ~~header~~ will input the presudo codes into the library and the OS system X expands source code using preprocessor directives
2	Compiler	.c	The compiler will translate statements into computer codes X translates high level language into low level ~~machine~~ assembly
3	Assembly	.o ~~.exe~~	It will assemblge the computerised codes, ~~pre-processor~~ library, input and output system X translates assembly into machine code
4	linker	.exe	It will link all the functions, statements, operator together before the programe is executed. X merges multiple object files into executable

-12

2. Stages of the compilation

A compiler program translates a high-level language code into a low-level language. With numbers 1–4 indicate in the right column the order in which the translation stages are performed:
(2 marks each; 8 marks total)

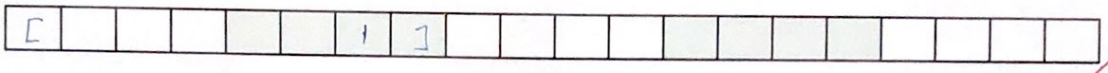
- Syntax analysis — 2
- Lexical analysis — 1
- Optimization and code generation — 4
- Semantic analysis — 3

3. Using `printf()`

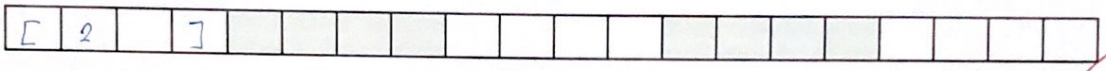
Given four variables defined and initialized as shown below, write down the exact printout of each statement. Use one character per cell. Write **NC** if the line does not compile at all. *(4 marks each; 32 marks total)*

```
int i = 1; short int s = 2; unsigned int u = 3; float f = 4.56f;
```

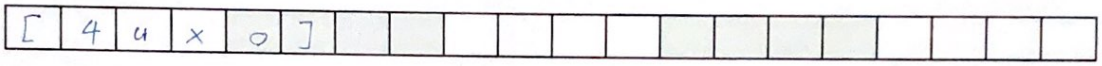
a) `printf("[% 6i]", i);`



b) `printf("[%-2i]", s);`

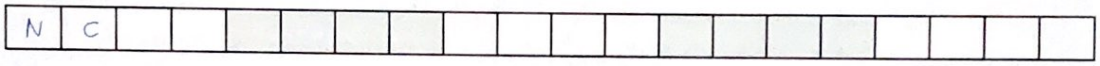


c) `printf("[-4%iuxo]", i);`



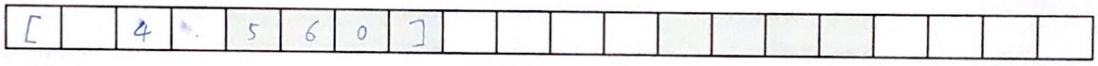
X-2

d) `printf("[%-+6.4i]", s);`



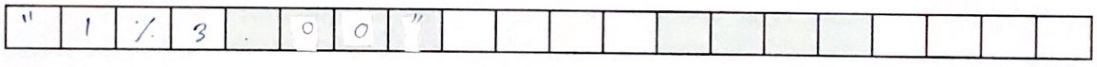
X-4

e) `printf("[%+3.3f]", f);`



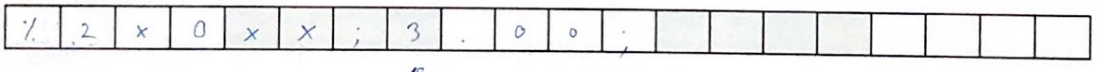
X-1

f) `printf("\"%-1i%%%4.4u\"", i, u);`



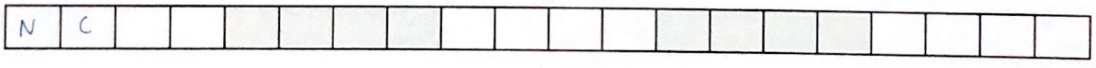
X-4

g) `printf("%%2x0x%x;%1.2f;", u, f);`



X-4

h) `printf(\"0%o%+0.3x\", s * 10, u);`



4. Using `scanf()`

Given the same four variables as above, separately for each example, for each call to `scanf()` and user input write down the result of the call and values of individual variables after the call returns. Write **NC** if the line does not compile at all. *(6 marks each row, 24 marks total)*

#	Function call	User input	Result	i	s	u	f
a)	`scanf("%f-%x", &f, &u);`	`9.87-ff`	%9.87ff	X	X	ff	9.87
b)	`scanf("%o.%f:%i", &u, &f, &i);`	`10.7.10`	%10.7:10	X10	X	10	7
c)	`s=(short)scanf("%4o%2i",&u,&i);`	`000123`	000123	23	X	0001	X
d)	`scanf("%3f%*u%i%%", &f,*&i);`	`1e20-0%`	3f*u	0	X	X	1e20

End of quiz.

-19