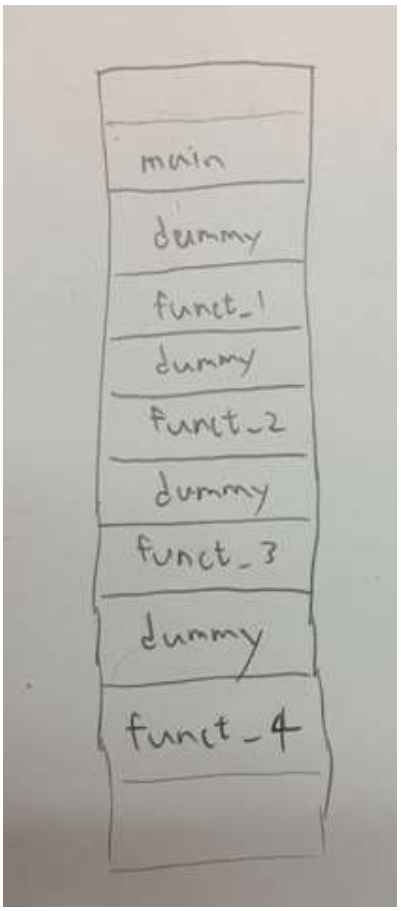
B07902068 陳柏豪 資工二

HW3 Report

a.

Pointer addresses showed by gdb are as following.
main: rbp = 0x7fffffffe700, rsp = 0x7fffffffe6f0
dummy: rbp = 0x7fffffffe6e0, rsp = 0x7fffffff4a80
funct_1: rbp = 0x7fffffff4a70, rsp = 0x7fffffff4a50
dummy: rbp = 0x7fffffff4a40, rsp = 0x7ffffffeade0
funct_2: rbp = 0x7ffffffeadd0, rsp = 0x7ffffffeadb0
dummy: rbp = 0x7ffffffeada0, rsp = 0x7ffffffe1140
funct_3: rbp = 0x7ffffffe1130, rsp = 0x7ffffffe1110
dummy: rbp = 0x7ffffffe1100, rsp = 0x7ffffffd74a0
funct_4: rbp = 0x7ffffffd7490, rsp = 0x7ffffffd7470



b.

Yes.
Since we didn't change its value outside the function, it will be the same.
If we change its value outside the function and compile the program without optimization, the value won't be the same.
If we change its value outside the function and compile the program with optimization, the value will be the same as when calling setjmp(), since it's stored in register.

c.

The usage of dummy is to prevent the stack frame of funct_1, funct_2, funct_3 and funct_4 from being overwritten by signal handler, scheduler or any other called function.

d.

No, stack smashing will be detected, and the process will be terminated.
Since main will call scheduler which will overwrite the stack frame of the fist dummy, the first dummy will lose its data and is unable to return.

Result showed by gdb is as following.

```
Breakpoint 3, dummy (name=1) at hw3.c:24
24          {
(gdb) print $rbp
$19 = (void *) 0x7fffffffe6e0

Breakpoint 7, 0x00005555555552fc in Scheduler ()
(gdb) print $rbp
$35 = (void *) 0x7fffffffe6e0
```

We see that the rbs are the same.

e.

I finish my program by first dealing with task1 and task2 and understanding stack frame during the process. Then, implement task3 and understanding how to handle signals. I think it a good thing that I know how to use gdb to see the stack frame of my program through this homework.