

1093319 魏博彥 計組 project1:

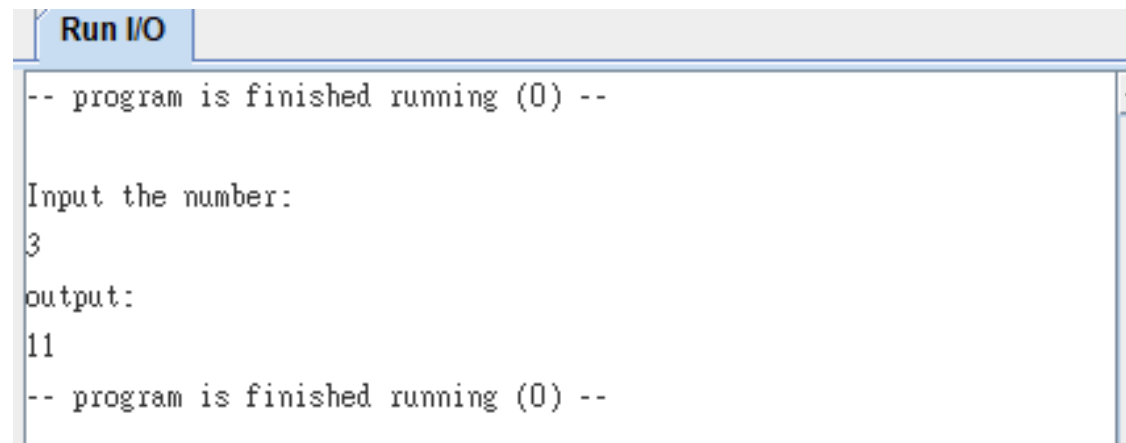
我自己有另外寫一個 C code 做對照用，有一併附上了

- Input=3

#C++

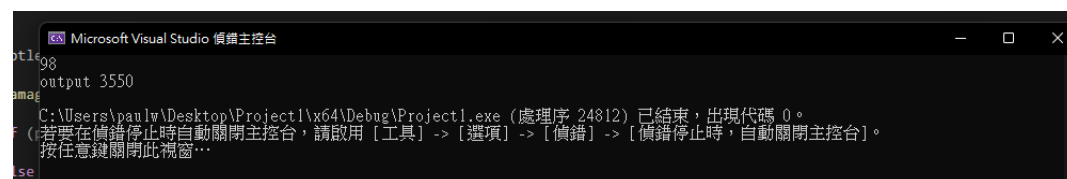


Risc-v:

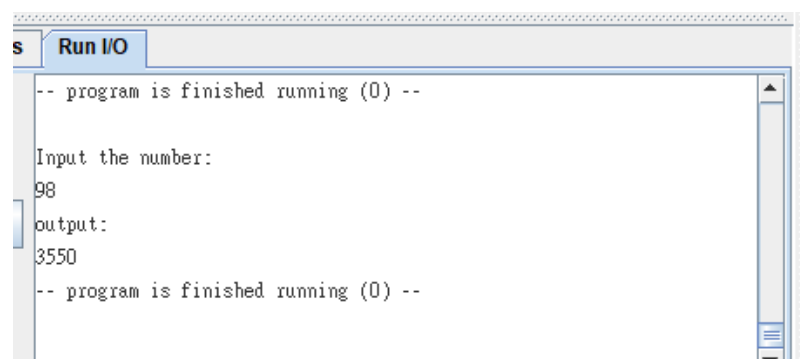


- Input=98

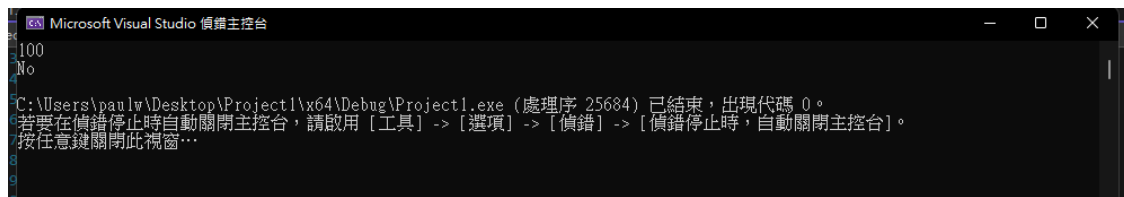
C++:



Risc-v:

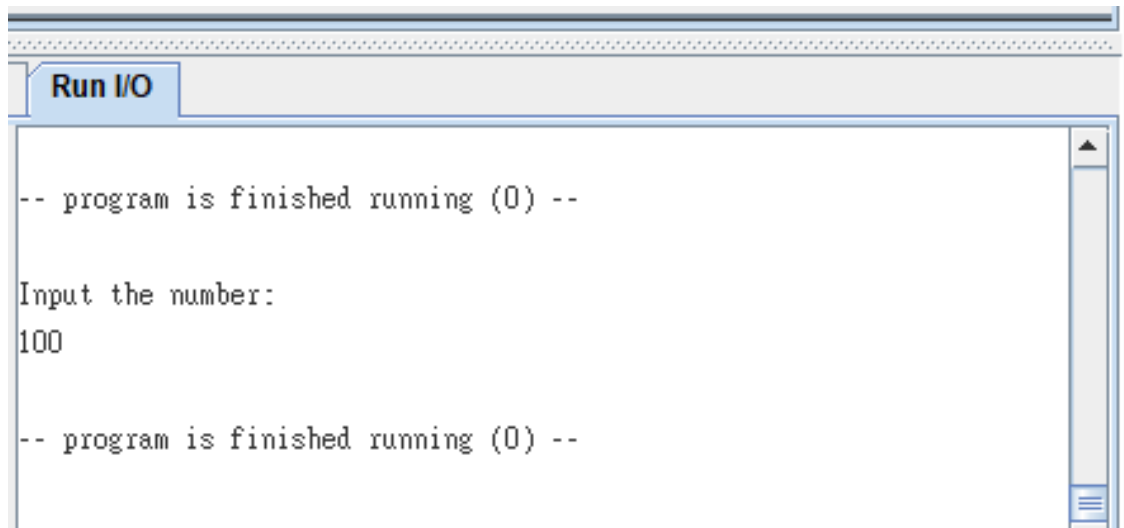


C++:



```
Microsoft Visual Studio 偵錯主控台
100
No
C:\Users\paulw\Desktop\Project1\Debug\Project1.exe (處理序 25684) 已結束，出現代碼 0。
若要在偵錯停止時自動關閉主控台，請啟用 [工具] -> [選項] -> [偵錯] -> [偵錯停止時，自動關閉主控台]。
按任意鍵關閉此視窗...
```

Risc-v:



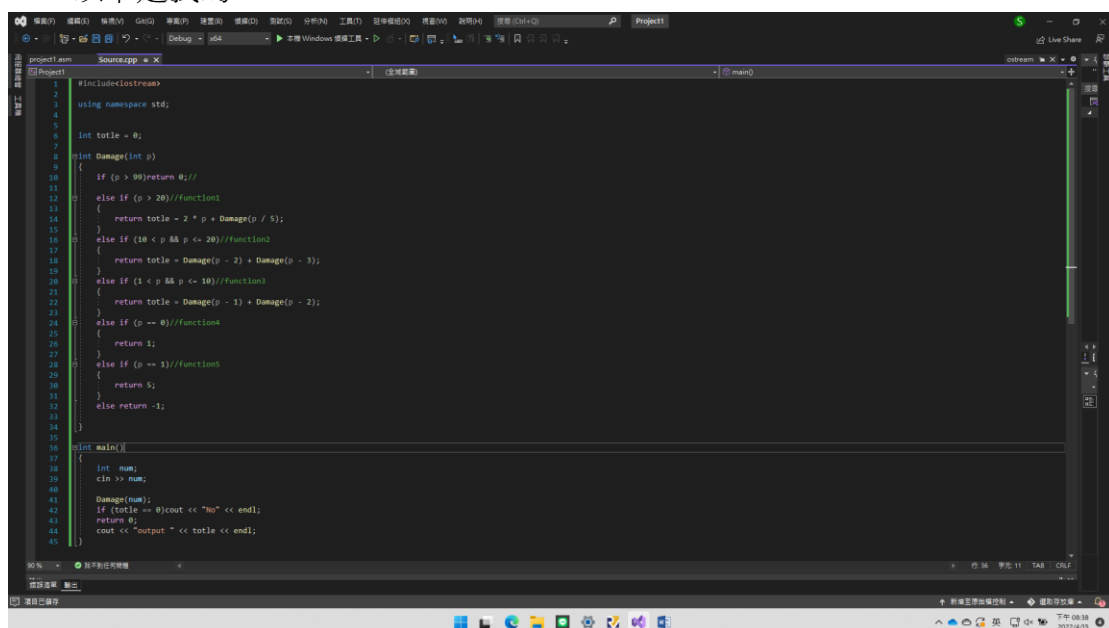
```
Run I/O

-- program is finished running (0) --

Input the number:
100

-- program is finished running (0) --
```

● 以下是我的 C++ code:



```
Source.cpp
1 #include <iostream>
2
3 using namespace std;
4
5 int totle = 0;
6
7 int Damage(int p)
8 {
9     if (p > 0) return 0;
10
11     else if (p > 20) //function1
12     {
13         return totle = 2 * p + Damage(p / 5);
14     }
15
16     else if (10 < p && p <= 20) //function2
17     {
18         return totle = Damage(p - 2) + Damage(p - 3);
19     }
20
21     else if (1 < p && p <= 10) //function3
22     {
23         return totle = Damage(p - 1) + Damage(p - 2);
24     }
25
26     else if (p == 0) //function4
27     {
28         return 1;
29     }
30
31     else if (p == 1) //function5
32     {
33         return 5;
34     }
35
36     else return -1;
37 }
38
39 int main()
40 {
41     int num;
42     cin >> num;
43
44     Damage(num);
45     if (totle == 0) cout << "No" << endl;
46     return 0;
47     cout << "output " << totle << endl;
48 }
```

1. 我是先將要用到的數字都先設好值，並且答案一開始是 0。
2. 在 input 之後呼叫 fun0。

```

1  .globl main
2  .data
3      input:  .string "Input the number:\n"
4      output: .string "output:\n"
5  .text
6
7  main:
8      addi s2,x0,100#比大小用的數字
9      addi s3,x0,20#s3=20
10     addi s4,x0,10#s4=10
11     add s5,x0,x0#final ans
12     addi s9,x0,5#輔助計算用的
13     addi s6,x0,1
14     la a0,input#這邊開始是input
15     li a7,4
16     ecall
17     li a7,5
18     ecall
19     add s7,a0,x0#這邊表s7=input
20     jal ra,fun0

```

3. Fun0 的作用是比较目前值的大小去跳到對應的 case 的 function，並且如果值>99 則直接跳到 Exit 結束。

```

31
32 fun0:
33     bge s7,s2,Exit#input>99 exit
34     beq s7,s6,fun5#input==1
35     beq s7,x0,fun4#input==0
36     beq s7,s4,fun3#input==10
37     beq s7,s3,fun2#input==20
38     bge s7,s3,fun1#20<input
39     bge s7,s4,fun2#10<input
40     bge s7,s6,fun3#1<input
41     blt s7,x0,fun6#else

```

4. Fun1 對應到的就是我 c code 的//function1 的部分，p>20。

1093319-HW1	riscv2.asm	riscv3.asm
42	fun1:#p > 20	
43	addi sp,sp -4#留出一個空間	
44	sw ra,0(sp)	
45	add t3,s7,x0	
46	slli t3,t3,1#2*p	
47	add s5,s5,t3	
48	div s7,s7,s9#p/5	
49	jal ra,fun0#go back to the fun0 => recursive	
50	lw ra,0(sp)	
51	addi sp,sp,4#把空間還回去	
52	jalr x0,0(ra)	

5. Fun2 與 fun3 分別對應到的是($10 < p \leq 20$)及($1 < p \leq 10$)的 case
 這邊 stack 留的位置比 fun1 多一個是因為傳入的值(在我的 c code 中是 p)會用到兩次，第一次使用的時候會動到它裡面值所以在下一次用到前，藉由事先保存的值對他做更新，使他變回一開始傳進來時的值。

```

1093319.HW1 riscv2.asm riscv3.asm
51      addi sp,sp,4#把空間還回去
52      jalr x0,0(ra)
53      fun2:#10 < p && p <= 20
54      addi sp,sp,-8#給出兩個位置 因為傳入的值在這邊要用到兩次
55      sw s7,0(sp)
56      sw ra,4(sp)
57      addi s7,s7,-2#p-2
58      jal ra,fun0#go back to the fun0 => recursive
59      lw s7,0(sp)#因為前面已經對s7中的值做過更改 所以要把它更新回來
60      addi sp,sp,4#因為用不到了
61      addi s7,s7,-3#p-3
62      jal ra,fun0#go back to the fun0 => recursive
63      lw ra,0(sp)
64      addi sp,sp,4
65      jalr x0,0(ra)
66      fun3:#1 < p && p <= 10
67      addi sp,sp,-8
68      sw s7,0(sp)
69      sw ra,4(sp)
70      addi s7,s7,-1#p-1
71      jal ra,fun0#go back to the fun0 => recursive
72      lw s7,0(sp)
73      addi sp,sp,4
74      addi s7,s7,-2#p-2
75      jal ra,fun0#go back to the fun0 => recursive
76      lw ra,0(sp)
77      addi sp,sp,4
78      jalr x0,0(ra)

```

6. 最後是($p=4$)、($p=1$)以及 otherwise 的 case。
 因為直接就是回傳值了所以把數字加到存答案的 s5 中。

```

79      fun4:#p == 0
80      addi s5,s5,1
81      jalr x0,0(ra)
82      fun5:#p == 1
83      addi s5,s5,5
84      jalr x0,0(ra)
85
86      fun6:#else
87      addi s5,s5,-1
88      jalr x0,0(ra)
89

```

7. 最後是($p \geq 99$)的時候直接結束。
 8. Output 輸出答案(s5)，然後結束程式。

```

21      la a0,output
22      li a7,4
23      ecall
24      mv a0,s5
25      li a7,1
26      ecall
27      j Exit
28      Exit:
29      li a7,10
30      ecall

```