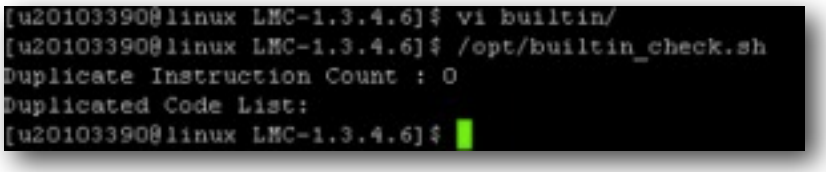
1. LMC OP Code를 개인화 하시오.

=> 다음 표는 Operation Code를 개인화 한 후 작성한 Manual 입니다.

	[1]			[2]			[3]		
[ ][1]	산술연산	덧셈	OORR	산술연산	•	•	기타	SHL	OORR
[ ][2]			OORMD		곱셈	OORMD		SHR	OORR
[ ][3]			OORMFA			OORMFA		RTL	OORR
[ ][4]		뺄셈	OORR		•	•		RTR	OORR
[ ][5]			OORMD		나눗셈	OORMD		SHRA	OORR
[ ][6]			OORMFA			OORMFA	•	•	•
[ ][7]	논리연산	OR	OORR	논리연산	AND	OORR	논리연산	XOR	OORR
[ ][8]			OORMD			OORMD			OORMD
[ ][9]			OORMFA			OORMFA			OORMFA

	[4]			[5]			[6]		
[ ][1]	LOAD	DBOX	OORR	STORE	DBOX	OORR	cal/ret	JCALL	OORMFA
[ ][2]			OORMD			OORMD			OORMD
[ ][3]			OORMFA			OORMFA		JSUB	OORMFA
[ ][4]		HALF	OORR		HALF	OORR			OORMD
[ ][5]			OORMD			OORMD		RSUB	OORMFA
[ ][6]			OORMFA			OORMFA			OORMD
[ ][7]	CMP		OORR	JMP		OORR		CALL	OORMFA
[ ][8]			OORMFA			OORMD			OORMD
[ ][9]			OORMD			OORMFA	•	•	•

[9][ ]	IMUL	92XX	OORMD	•	•	•
	IDIV	93XX	OORMFA	DIV	986X	OOOR
	IMUL	94XX	OORMD	MUL	993X	OOOR
	IDIV	95XX	OORMFA	NEG	984X	OOOR
	IMUL	994X	OOOR	NOT	985X	OOOR
	IDIV	995X	OOOR	SKIP3	990X	OOOR
	PUSH	988X	OOOR	JCALL	981X	OOOR
	POP	989X	OOOR	JRET	9998	OOOO
	JSUB	982X	OOOR	CALL	983X	OOOR
	RSUB	992X	OOOR	RET	9999	OOOO



=> linux.cs.kookmin.ac.kr 계정에 업데이트 후 중복 확인하여 이상 없었습니다.

2. 다음 프로그램을 OoAa Type 명령어를 사용하지 말고 확장된 명령어들(OoRR, OoRMFA, OORMD)의 개인화된 코드로 다시 작성하여 실행 하시오.

=> 제시된 코드는 입력받은 두 수를 뺀 값의 절대값을 출력하는 코드입니다.

```
IN    10   // 510
STA   11   // 211
IN    10   // 510
STA   12   // 212
SUBA  11   // 411
SKN        // 803
JMP   09   // 909
LDA   11   // 111
SUBA  12   // 412
OUT   12   // 612
COB        // 700
BOX   0    // 000
BOX   0    // 000
```

LMC			Comment
Addr NO.	code	mnemonic	
0	510	IN 10	입력
1	211	STA 11	11번지에 저장(A)
2	510	IN 10	입력
3	212	STA 12	12번지에 저장(B)
4	411	SUBA 11	B-A (Accum.REG = B)
5	803	SKN	음수이면 스킵
6	909	JMP 9	점프
7	111	LDA 11	A.REG에 A값 로드
8	412	SUBA 12	A-B (Accum.Reg = A)
9	612	OUT12	출력
10	700	COB	홀트
11	000		(A)값 저장
12	000		(B)값 저장
•	•	•	
•	•	•	

Personalized Extended LMC								Comment
Label	Addr.NO.			format	code			mnemonic
	0	1	2	OORMFA	4600	0100	9200	LD_A 9200 0100	입력
	3	4	5	OORMFA	5300	0011	0000	ST A 33	33번지에 저장(A)
	6	7	8	OORMFA	4600	0100	9200	LD_A 9200 0100	입력
	9	10	11	OORMFA	5300	0012	0000	ST A 36	33번지에 저장(B)
	12	13	14	OORMFA	1600	0056	0000	SUB A 33	B-A (Accum.REG = B)
	15	16	17	OOOR	9903	0050	0000	SKP3 3	음수이면 3box 스킵
	18	19	20	OORMFA	5900	0000	0000	JMP LABEL_1	27번지로 점프
	21	22	23	OORMFA	4600	0036	0000	LD A 33	A.REG에 A값 로드
	24	25	26	OORMFA	1600	0053	0000	SUB A 36	A-B (Accum.Reg = A)
label_1	27	28	29	OORMFA	5300	0053	0000	ST_A 9200 0120	출력
	30	31	32	OORR	0700	0000	0000	COB	홀트
	33	34	35	•	0000	0000	0000		(A)값 저장
	36	37	38	•	0000	0000	0000		(B)값 저장
	39	40	41	•	•	•	•	•	

3. Sum(1..n) 프로그램을 3개의 함수로 나누고 main에서는 이 3개 함수를 호출하는 구조로 바꾸시오

=>이 전에 작성했던 프로그램 코드를 여러 함수로 나누어 보았습니다.

Extended LMC									Comment
Label	Addr. NO.			format	code			mnemonic	
#	0	1	2	#	0001	0000	0000	NOP	상수선언 (&0 = 1, &1 = 0)
main	3	4	5	OORMFA	1330	0001	0000	LD X 1	X.Reg 를 0으로 초기화
	6	7	8	OORMFA	1300	0100	9200	LD_A 9200 0100	A.Reg 에 n입력
BgnLoop	9	10	11	OORR	3230	0000	0000	ADD X A	X.Reg에 A.reg 값 더해 넣기
	12	13	14	OORMFA	4300	0000	0000	SUB A 0	A.Reg-1
	15	16	17	OOAA	9821	0000	0000	SKP3 1	조건문 : 0 이 되면 연산종료
	18	19	20	OORMD	7300	0009	0000	JMP BgnLoop	9번 주소로 이동
OUT	21	22	23	OORMFA	2230	0120	9200	ST_X 9200 0120	X.Reg 출력.
	24	25	26	OOAA	700	0000	0000	COE 0	홀트
.	.	.	.	.	.	.	.	.	.

* 13. 10. 2 일 제출한 “SUM(1•••n)” 코드입니다.

=> 다음의 표는 C++로 표현한 의사(pseudo) 코드 입니다.

```
#define A0 = 0
#define A1 = 1

static int A_Reg, X_reg;

void main() {

     X_Reg = A0;
     cin>>A_Reg;

// Label : BgnLoop  (box 9)
     do {
          X_Reg+=A_Reg;
          A_Reg--;
     } while ( !(A_Reg==0) )

//Label : OUT  (box 21)
     cout<<X_Reg;

     return 0; // COE
}
```

```
#define A0 = 0
#define A1 = 1

static int A_Reg, X_reg;

void main() {
     SetN();
     SumN();
     Print();
     return 0; // COE
}

void SetN() {
     X_Reg = A0;
     cin>>A_Reg;
}

void SumN() {
// Label : BgnLoop
     do {
          X_Reg+=A_Reg;
          A_Reg--;
     } while ( !(A_Reg==0) )
}

void Print(){cout<<X_Reg;}
```

Personalized Extended LMC									Comment
Label	Addr. NO.			format	code			mnemonic	
#	0	1	2	#	0001	0000	0000	NOP	상수선언 (&0 = 1, &1 = 0)
main	3	4	5	OORMFA	6100	0018	0000	JCALL SetN	18번지로 이동
	6	7	8	OORMFA	6100	0030	0000	JCALL SumN	30번지로 이동
	9	10	11	OORMFA	6100	0054	0000	JCALL Print	54번지로 이동
	12	13	14	OORR	0700	0000	0000	COE	홀트
	15	16	17	•	0000	0000	0000	•	•
SetN	18	19	20	•	0000	0000	0000	•	•
	21	22	23	OORMFA	4330	0001	0000	LD X 1	X.Reg 를 0으로 초기화
	24	25	26	OORMFA	4300	0100	9200	LD_A 9200 0100	A.Reg 에 n입력
	27	28	29	OOOO	9998	0000	0000	JRET	메인으로 복귀
SumN	30	31	32	•	0000	0000	0000	•	•
BgnLoop	33	34	35	OORMFA	4330	0001	0000	LD X 1	X.Reg 를 0으로 초기화
	36	37	38	OORMFA	4300	0100	9200	LD_A 9200 0100	A.Reg 에 n입력
	39	40	41	OORR	1330	0000	0000	ADD X A	X.Reg에 A.reg 값 더해 넣기
	42	43	44	OORMFA	1400	0000	0000	SUB A 0	A.Reg-1
	45	46	47	OOAA	9821	0000	0000	SKP3 1	조건문 : 0 이 되면 연산종료
	48	49	50	OORMD	5800	0033	0000	JMP BgnLoop	33번 주소로 이동
	51	52	53	OOOO	9998	0000	0000	JRET	메인으로 복귀
Print	54	55	56	•	0000	0000	0000	•	•
	57	58	59	OORMFA	5330	0120	9200	ST_X 9200 0120	X.Reg 출력.
	60	61	62	OOOO	9998	0000	0000	JRET	메인으로 복귀
	63	64	65	•	•	•	•	•	•

* Op Code 개인화 후 여러 함수로 나눈 “SUM(1•••n)” 코드입니다.

4. 생각해보기

- Skip on Condition & JMP의조합을 어떤 Conditional Jump로 바꿀 것인가?

=> PSW 레지스터에 저장된 경우에 따라 스킵 바로 다음에 오는 점프 사용하는지를 보게 되는데, 이는 **C언어에서 if문**의 성질 (괄호 안의 논리결과가 False이면 if문 박스 안의 함수를 실행하지 않음) 과 동일합니다.

- 3번의 C 프로그램을 LMC 코드로 바꿀 때 상수, 변수, 파라미터, 반환값 등을 어떻게 표현 또는 처리 하였는가?

=> 상수는 변하지 않는 값이기에 코드 맨 앞줄에, 변수는 선언하지는 않았지만 필요에 따라 선언 해야 했다면 해당 함수 바로 뒤에 두었을 것입니다.
파라미터와 반환 값 또한 위 코드에서 선언하지는 않았지만 필요에 따라 선언 해야 했다면 수행하는 모든 함수 뒤 (여기선 63번지 이후가 될 것입니다.) 에 선언하여 같은 변수를 각 함수가 사용하게 하였을 것입니다.