Algorithm explanation

- First read in all the data. Numbers are separated by space and '\n'.
- Using simple `dfs` without pruning to search the answer, each page has two choices. Use the third spaces of each page in the data structure to note which chapter is chosen.
- At the exit of recursion, check if the chosen pages fit the requirement. That is realized by using a binary state compression.
- If the chosen mask equal to all in mask, then that means we find the correct answer, otherwise, just return to find the new path.
- If they fit the requirement, then just print the chosen chapters one by one and halt.

Essential parts of your code with sufficient comments

```
Init            LD      R6, StackAdd    ;

RdInN           JSR     Input           ;
                ST      R0, PageN       ;
                LD      R1, PageN       ;
                LD      R2, MasksAdd    ;
                LD      R5, PagesInfoAdd;
RdPagesInfo     ADD     R1, R1, #-1     ; loop
                BRn     ExitRdPagesInfo ; loop exit
                LD      R0, MaskAllIn   ;   \
                LDR     R3, R2, #0      ;   |
                ADD     R0, R0, R3      ;   | Do something to initialize the mask
all in.
                ST      R0, MaskAllIn   ;   |
                ADD     R2, R2, #1      ;   /
                JSR     Input           ; Read first chapter
                STR     R0, R5 #0       ;
                JSR     Input           ; Read second chapter
                STR     R0, R5, #1      ;
                ADD     R5, R5, #3      ; The tird is left for chosen.
                BRnzp   RdPagesInfo     ;
ExitRdPagesInfo LD      R0, PageN       ;
                JSR     dfs             ;

SolutionFound   LD      R2, PagesInfoAdd;
                LD      R1, PageN       ;
PrtChoosenPages ADD     R1, R1, #-1     ; loop
                BRn     Exit            ; loop exit
                LDR     R0, R2, #2      ;
                LDR     R0, R0, #0      ;
                JSR     PrintNumber     ;
                ADD     R2, R2, #3      ;
                BRnzp   PrtChoosenPages ;

Exit            HALT                    ;
; And other necessary things.
```

```
; dfs() ---------------------------------------------------
; find the solution
dfs             Do callee save.
                ADD     R0, R0, #-1     ; Next params value.
                BRn     dfsExit         ; Recursion exit.
                JSR     incSP           ; For chosenMask.
                LD      R3, PagesInfoAdd;
                ADD     R3, R3, R0      ; Actually it should be R5 + R0 - 1, but
I did R0-1 already.
                ADD     R3, R3, R0      ; Actually it should be R5 + R0 - 1, but
I did R0-1 already. (twice)
                ADD     R3, R3, R0      ; Actually it should be R5 + R0 - 1, but
I did R0-1 already. (trible)
                LDR     R1, R3, #0      ; First chapter.
                STR     R3, R3, #2      ; Choose First chapter.
                LD      R2, MasksAdd    ;
                ADD     R2, R2, R1      ;
                LDR     R2, R2, #-1     ; -1 referred to offset
                STR     R2, R6, #0      ; pass arguement
                LD      R2, MaskChosen  ;
                STR     R2, R6, #-1     ; Tmp.
                STR     R2, R6, #1      ; pass arguement
                JSR     incSP           ;
                JSR     incSP           ;
                JSR     OR              ;
                JSR     decSP           ;
                JSR     decSP           ;
                LDR     R2, R6, #0      ;
                ST      R2, MaskChosen  ;
                JSR     dfs             ; First dfs
                LDR     R1, R6, #-1     ;
                ST      R1, MaskChosen  ;
                ADD     R3, R3, #1      ; Move iterator.
                LDR     R1, R3, #0      ; Second chapter.
                STR     R3, R3, #1      ; Choose Second chapter.
                LD      R2, MasksAdd    ;
                ADD     R2, R2, R1      ;
                LDR     R2, R2, #-1     ; -1 referred to offset
                STR     R2, R6, #0      ; pass arguement
                LD      R2, MaskChosen  ;
                STR     R2, R6, #1      ; pass arguement
                JSR     incSP           ;
                JSR     incSP           ;
                JSR     OR              ;
                JSR     decSP           ;
                JSR     decSP           ;
                LDR     R2, R6, #0      ;
                ST      R2, MaskChosen  ;
                JSR     dfs             ; Second dfs
                JSR     decSP           ; For chosen mask.
                LDR     R1, R6, #0      ; recover
                ST      R1, MaskChosen  ;

dfsExit         JSR     CheckIsOK       ; Check whether it fits requirements.
                ADD     R0, R0, #0      ; Set cc.
```

```
                BRp     SolutionFound   ;
                Do callee save recover.
                RET
```

Questions TA asked you and your answer in Check

- How to optimize it?
- Before `dfs`, we should put the smaller chapter in one page before the lager one, and sort the pages according to the smaller one and according to the larger one if the smaller one is equal. After that, while performing `dfs`, we should return before meet end if we find that we found a number that we already picked, which means this path will never fit the requirement.