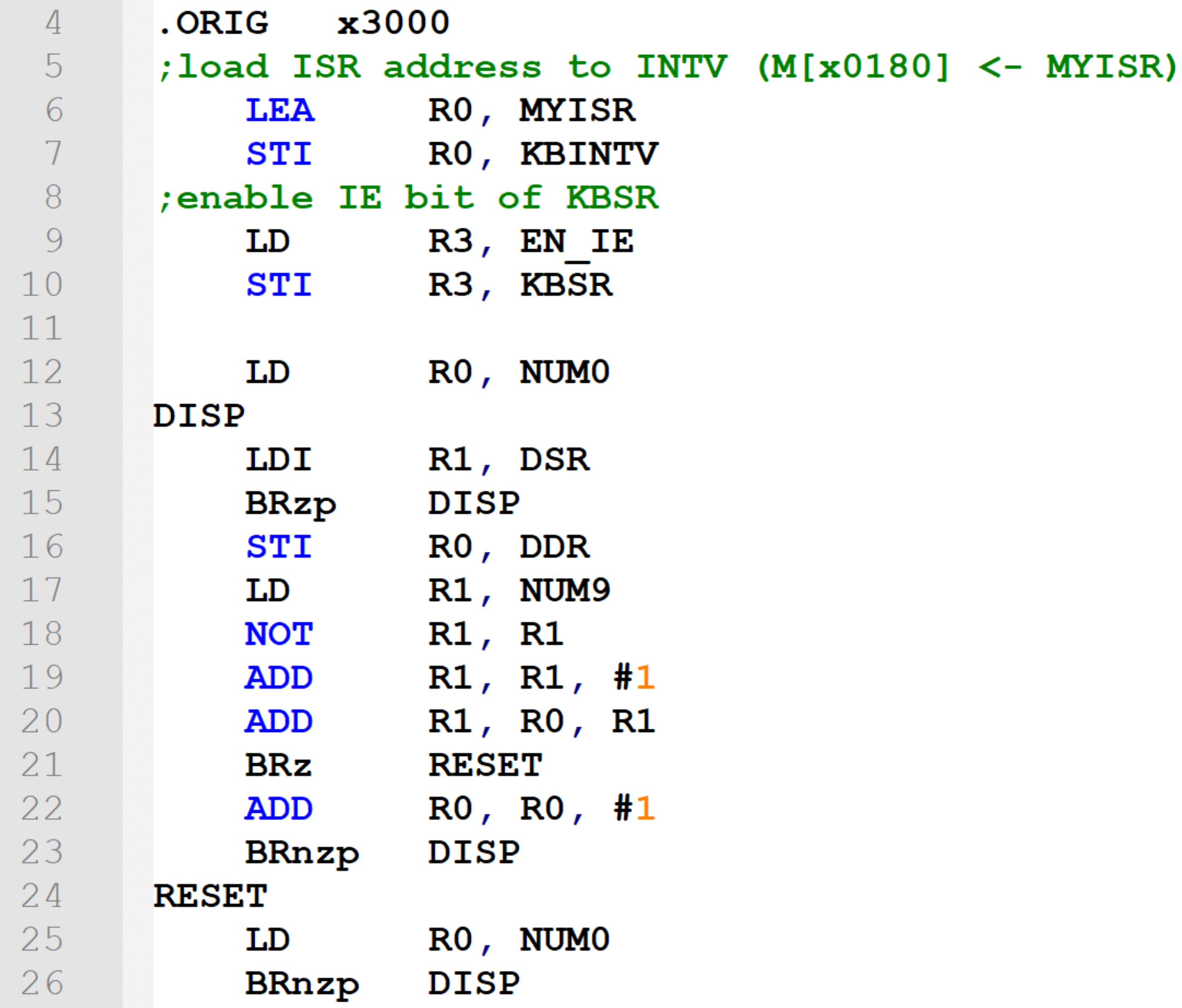
University of Illinois at Urbana-Champaign
Dept. of Electrical and Computer Engineering

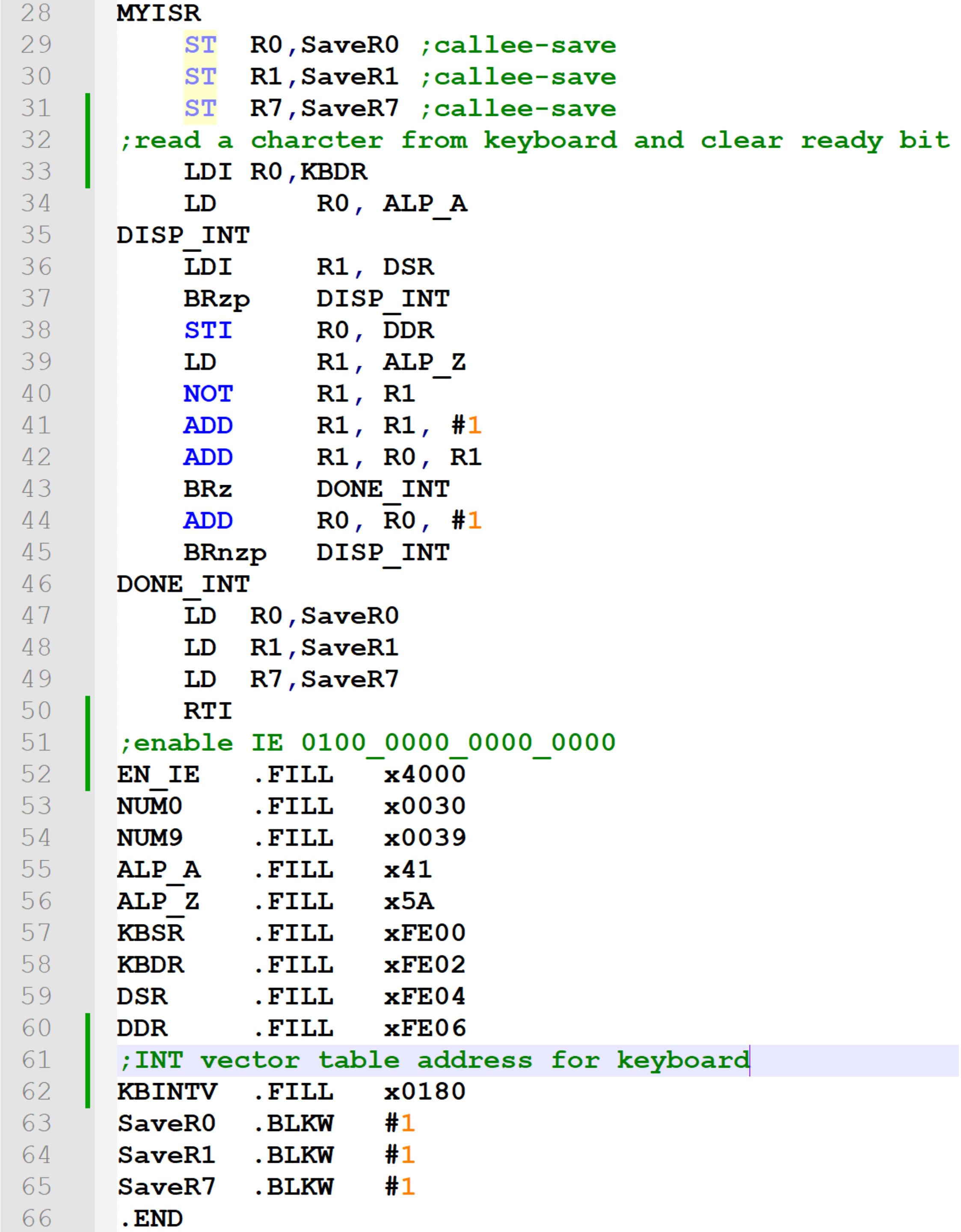
ECE 220: Computer Systems & Programming

Interrupt & Review1

Execute the Interrupt Code



```
.ORIG   x3000
;load ISR address to INTV (M[x0180] <- MYISR)
    LEA       R0, MYISR
    STI       R0, KBINTV
;enable IE bit of KBSR
    LD        R3, EN_IE
    STI       R3, KBSR

    LD        R0, NUM0
DISP
    LDI       R1, DSR
    BRzp      DISP
    STI       R0, DDR
    LD        R1, NUM9
    NOT       R1, R1
    ADD       R1, R1, #1
    ADD       R1, R0, R1
    BRz       RESET
    ADD       R0, R0, #1
    BRnzp     DISP
RESET
    LD        R0, NUM0
    BRnzp     DISP
```

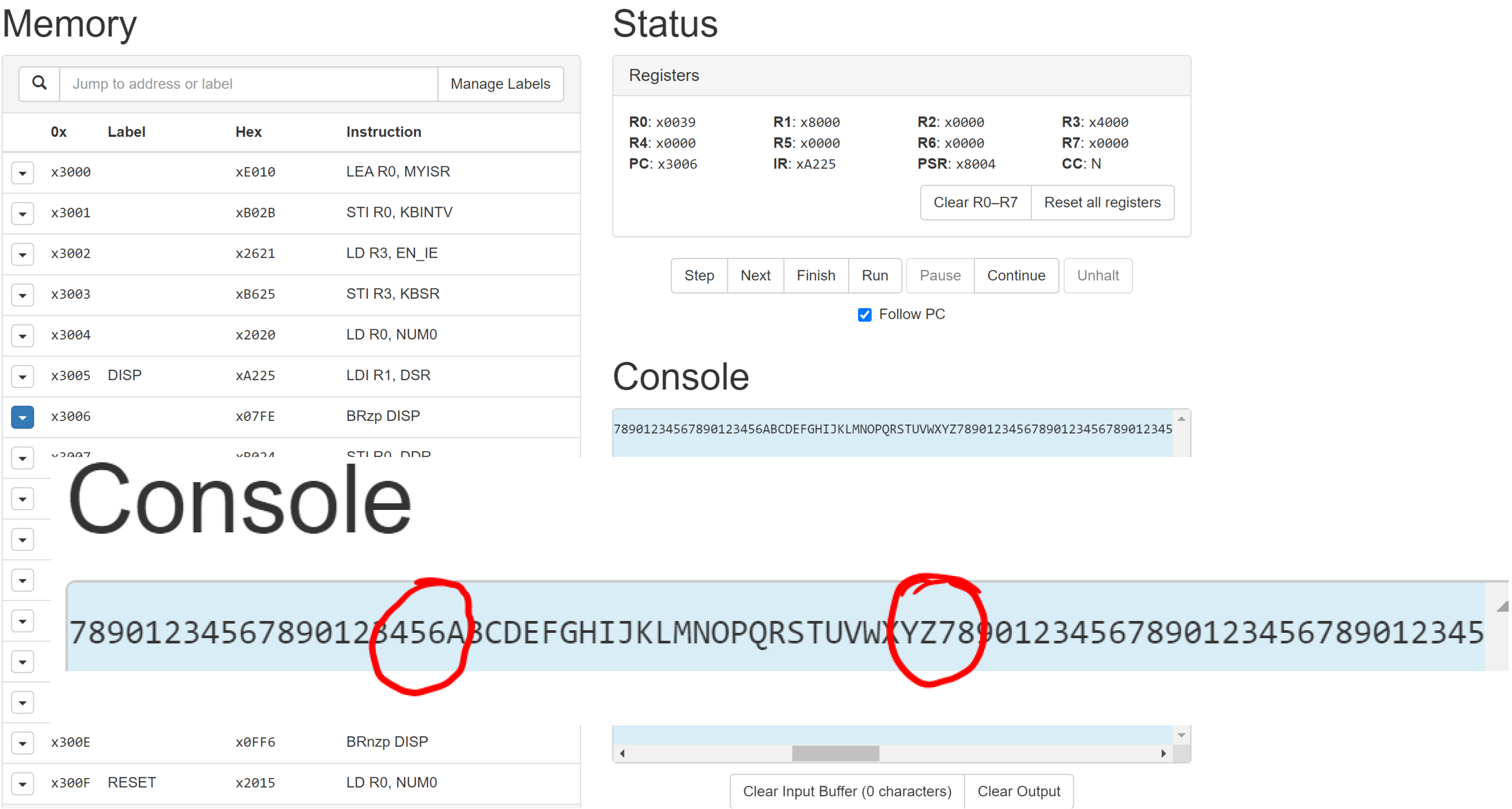


```
MYISR
    ST  R0,SaveR0 ;callee-save
    ST  R1,SaveR1 ;callee-save
    ST  R7,SaveR7 ;callee-save
;read a charcter from keyboard and clear ready bit
    LDI R0,KBDR
    LD        R0, ALP_A
DISP_INT
    LDI       R1, DSR
    BRzp      DISP_INT
    STI       R0, DDR
    LD        R1, ALP_Z
    NOT       R1, R1
    ADD       R1, R1, #1
    ADD       R1, R0, R1
    BRz       DONE_INT
    ADD       R0, R0, #1
    BRnzp     DISP_INT
DONE_INT
    LD  R0,SaveR0
    LD  R1,SaveR1
    LD  R7,SaveR7
    RTI
;enable IE 0100_0000_0000_0000
EN_IE     .FILL    x4000
NUM0      .FILL    x0030
NUM9      .FILL    x0039
ALP_A     .FILL    x41
ALP_Z     .FILL    x5A
KBSR      .FILL    xFE00
KBDR      .FILL    xFE02
DSR       .FILL    xFE04
DDR       .FILL    xFE06
;INT vector table address for keyboard
KBINTV    .FILL    x0180
SaveR0    .BLKW    #1
SaveR1    .BLKW    #1
SaveR7    .BLKW    #1
.END
```

****Assemble and Run the Code on the LC3 Web Simulator****

https://courses.grainger.illinois.edu/ece220/sp2020/lc3web/index.html

LC3 Web Simulator



Memory

0x	Label	Hex	Instruction
x3000		xE010	LEA R0, MYISR
x3001		xB02B	STI R0, KBINTV
x3002		x2621	LD R3, EN_IE
x3003		xB625	STI R3, KBSR
x3004		x2020	LD R0, NUM0
x3005	DISP	xA225	LDI R1, DSR
x3006		x07FE	BRzp DISP
x3007		xB024	STI R0, DDR
x300E		x0FF6	BRnzp DISP
x300F	RESET	x2015	LD R0, NUM0

Status

Registers

R0: x0039	R1: x8000	R2: x0000	R3: x4000
R4: x0000	R5: x0000	R6: x0000	R7: x0000
PC: x3006	IR: xA225	PSR: x8004	CC: N

☑ Follow PC

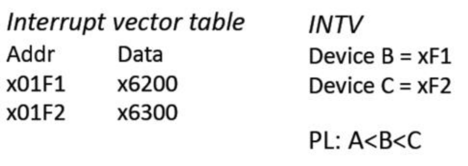
Console

```
789012345678901234 56ABCDEFGHIJKLMNOPQRSTUVWXYZ789012345678901234567890123 45
```

Console

```
789012345678901234 56ABCDEFGHIJKLMNOPQRSTUVWXYZ789012345678901234567890123 45
```

Nested Interrupts



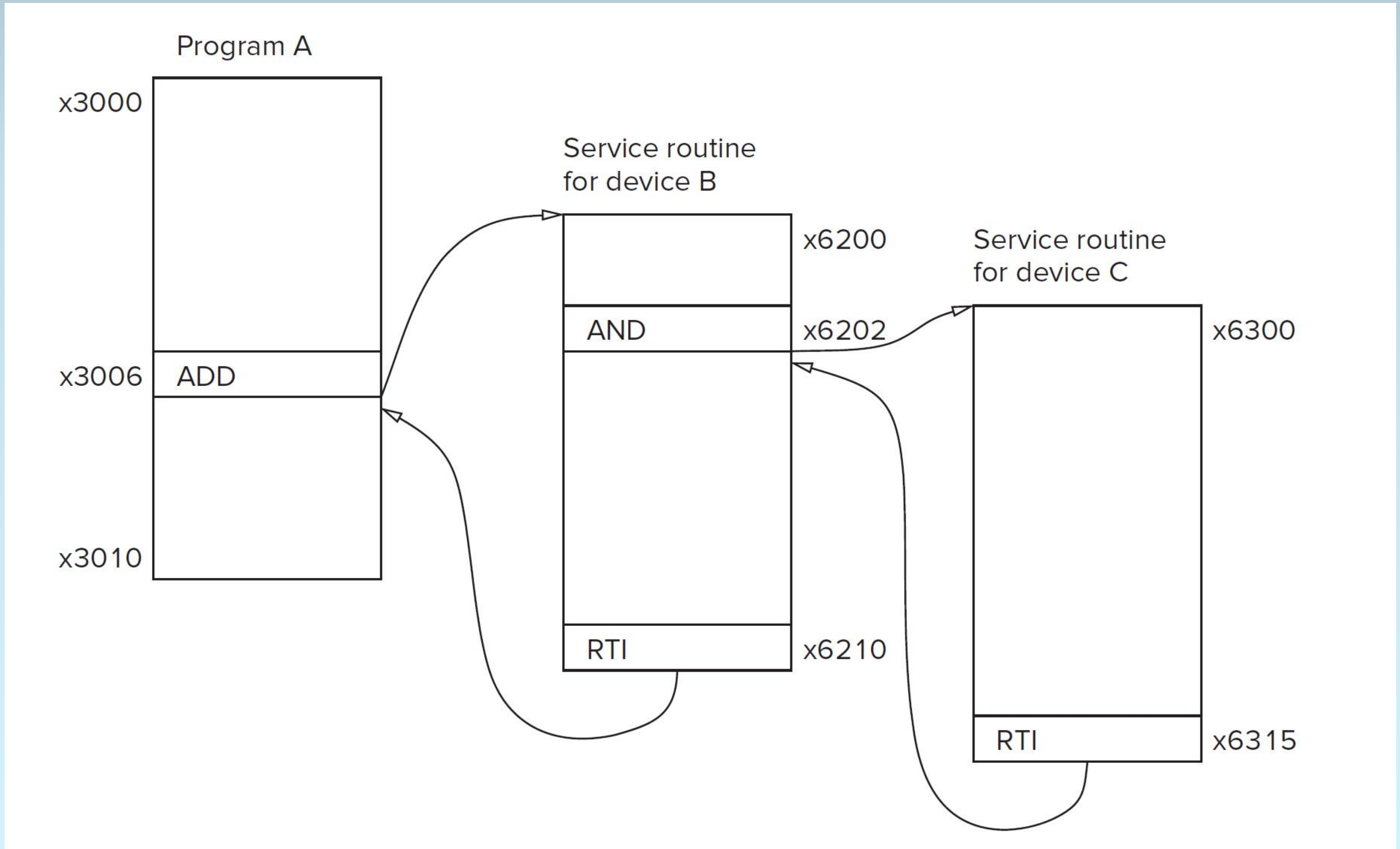
Interrupt vector table

Addr	Data
x01F1	x6200
x01F2	x6300

INTV

Device B = xF1
Device C = xF2

PL: A<B<C



Interrupt vector table

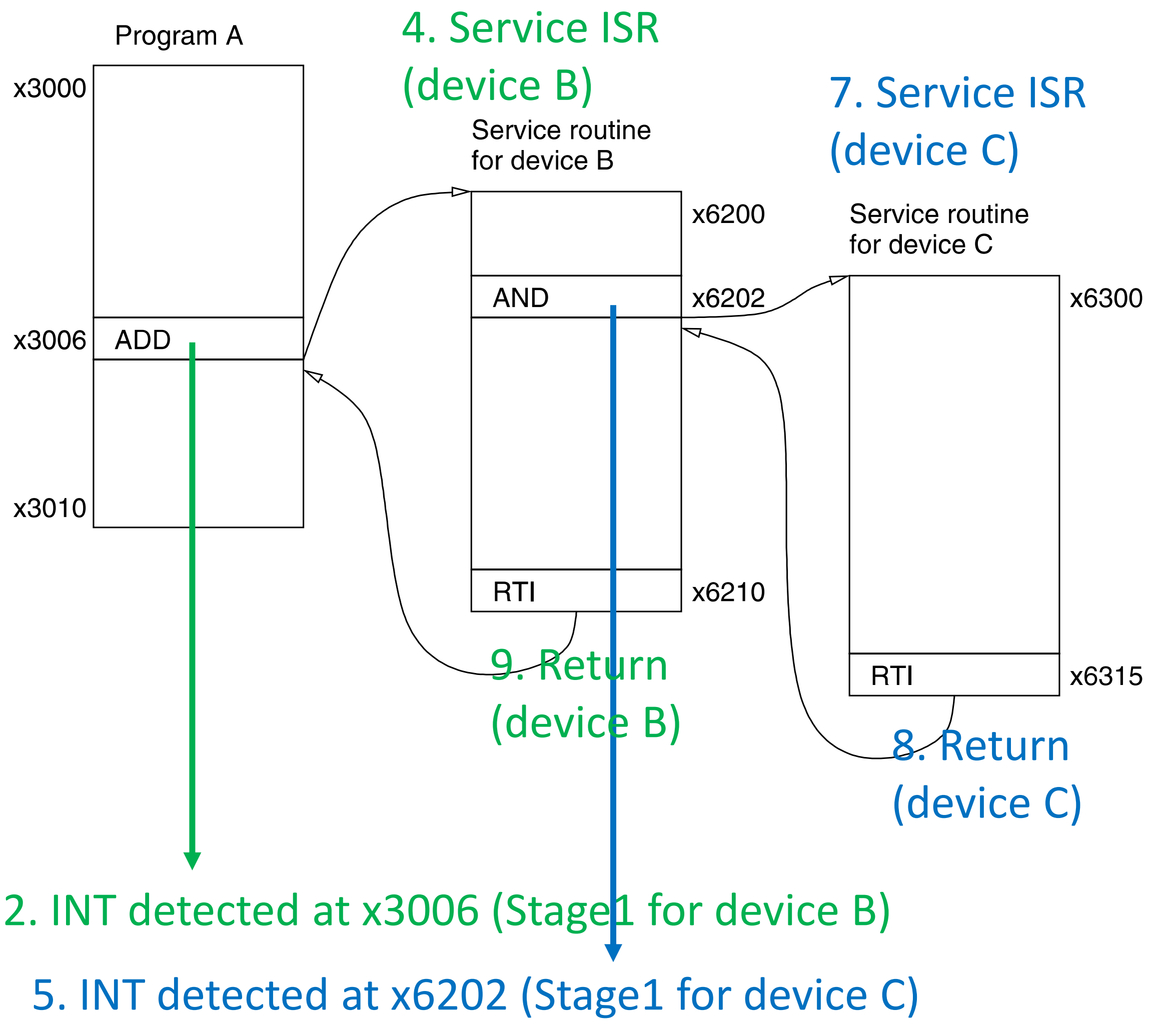
Addr	**Data**
x01F1	x6200
x01F2	x6300

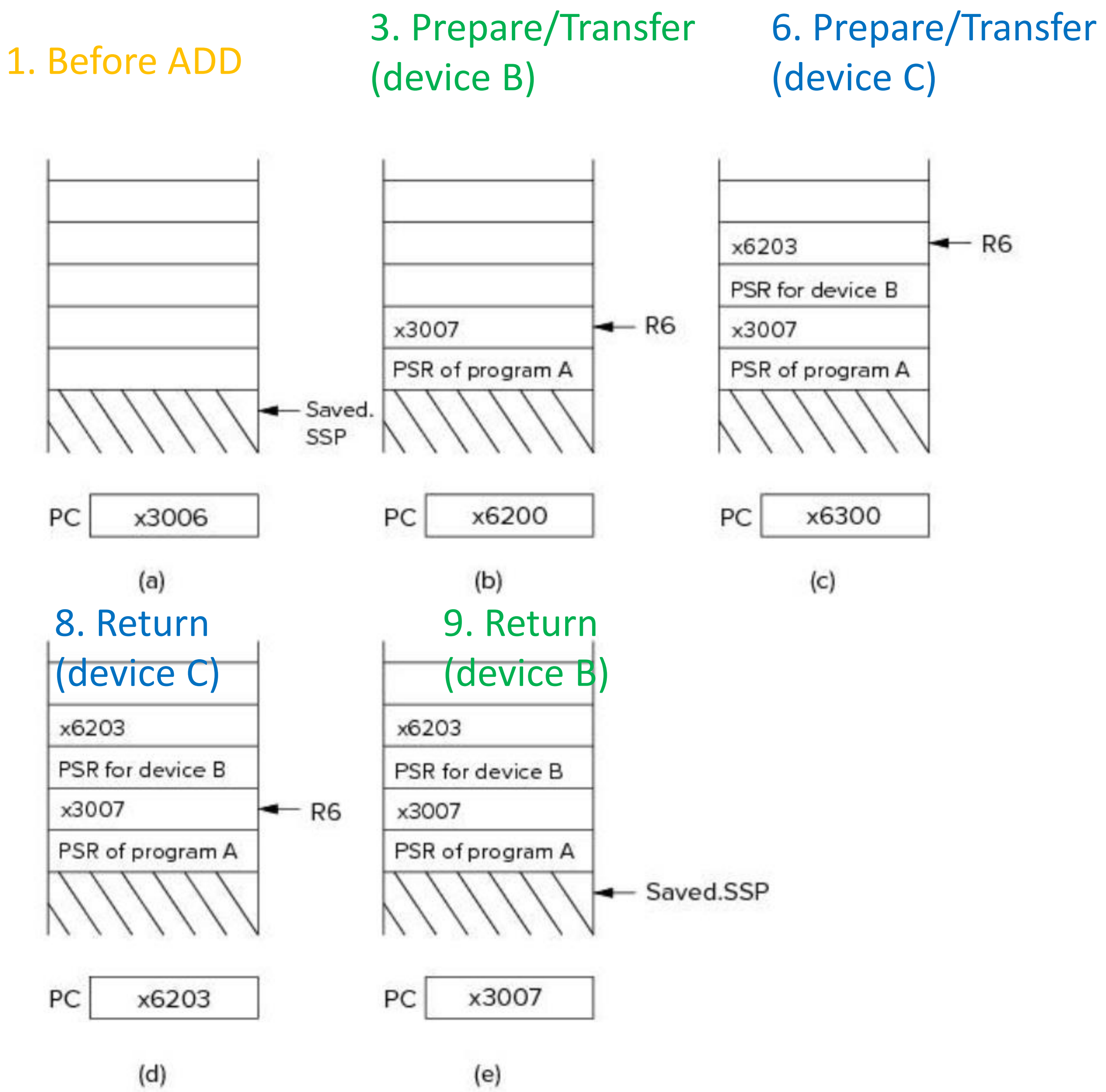
INTV

Device B = xF1
Device C = xF2

PL

A<B<C



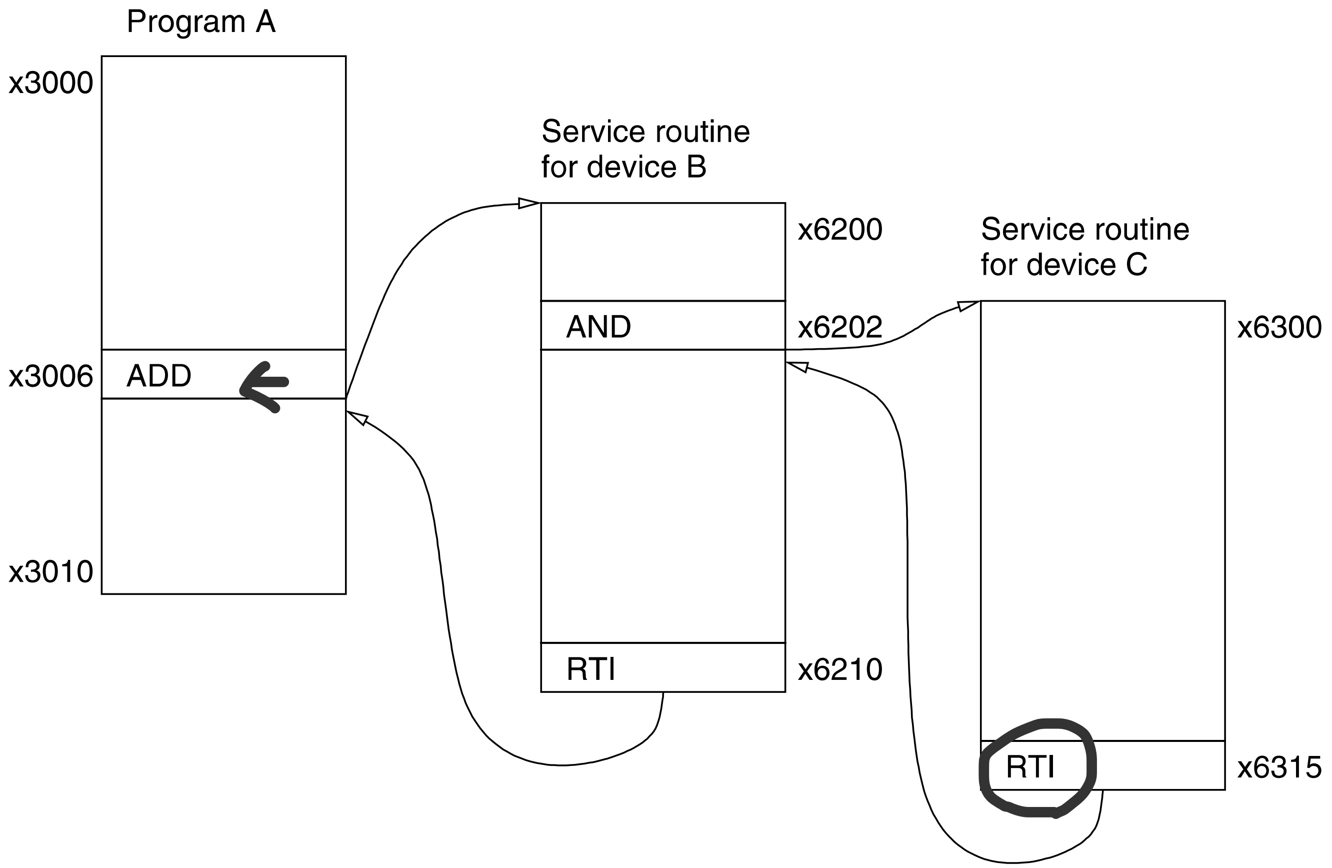


Q. Suppose a device A initiates an interrupt. The interrupt vector of device A is x30 and its ISR starts at x1200. What can you tell about the contents of any memory location?

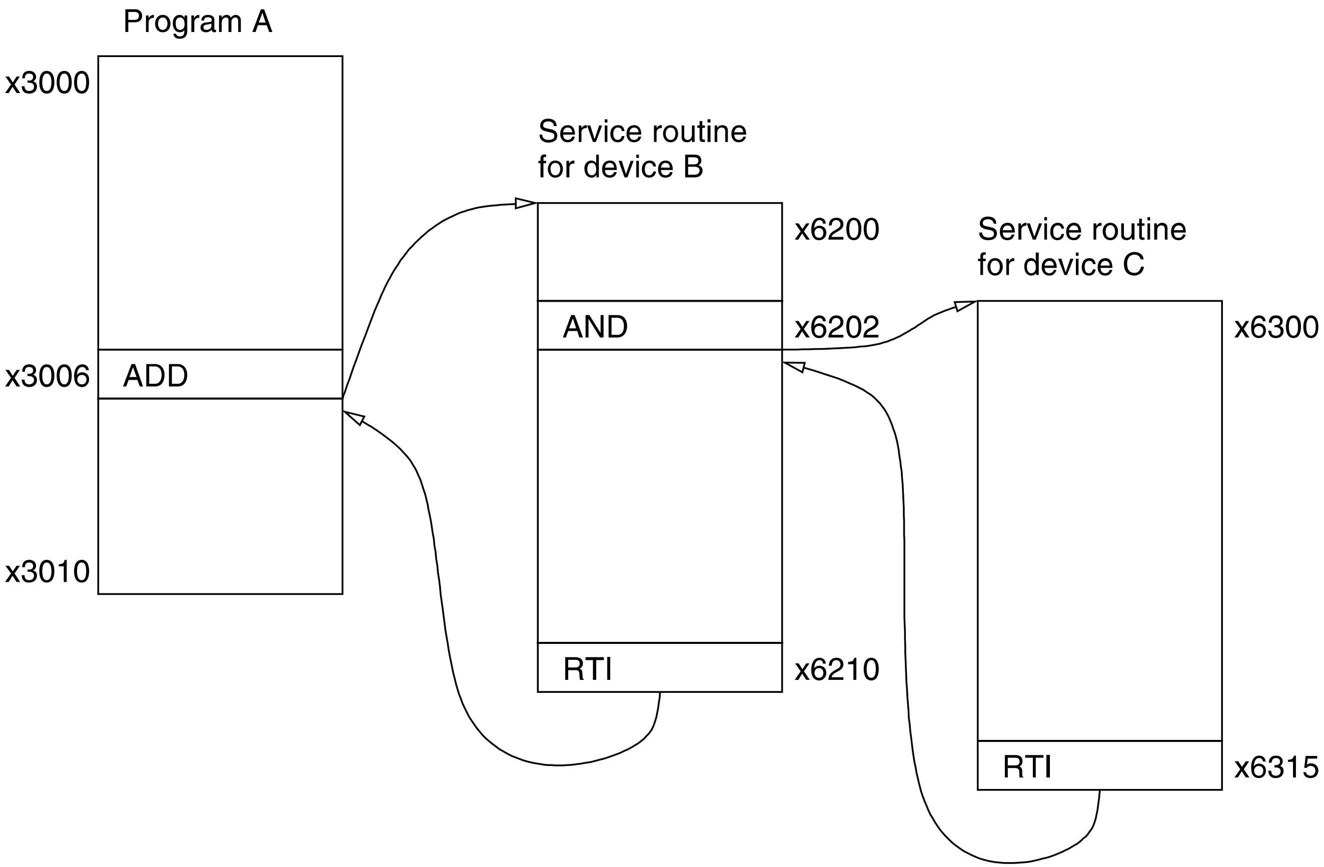
A. The content of address x0030 is x1200.
B. The content of address x0130 is x1200.
C. The content of address x1200 is x0030.
D. The content of address x1200 is x0130.
E. You cannot determine anything about the memory by the above information.

Q. After the RTI of device C is executed, what is the value of PC? Draw the Supervisor Stack and R6 (assume textbook-style stack).

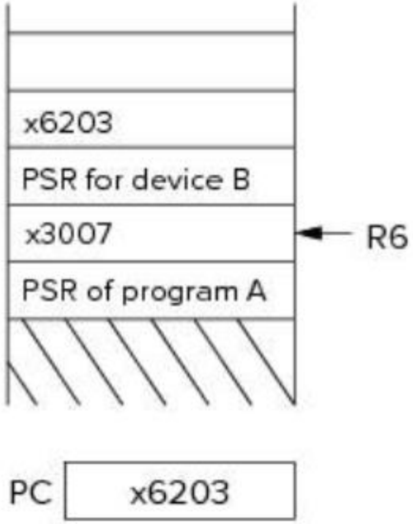
PC = x6203



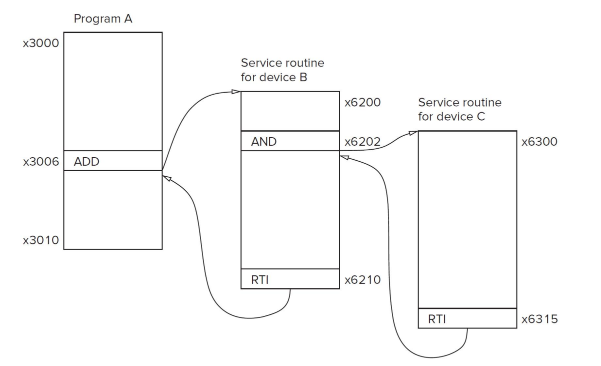
Q. After the RTI of device C is executed, what is the value of PC? Draw the Supervisor Stack and R6 (assume textbook-style stack).

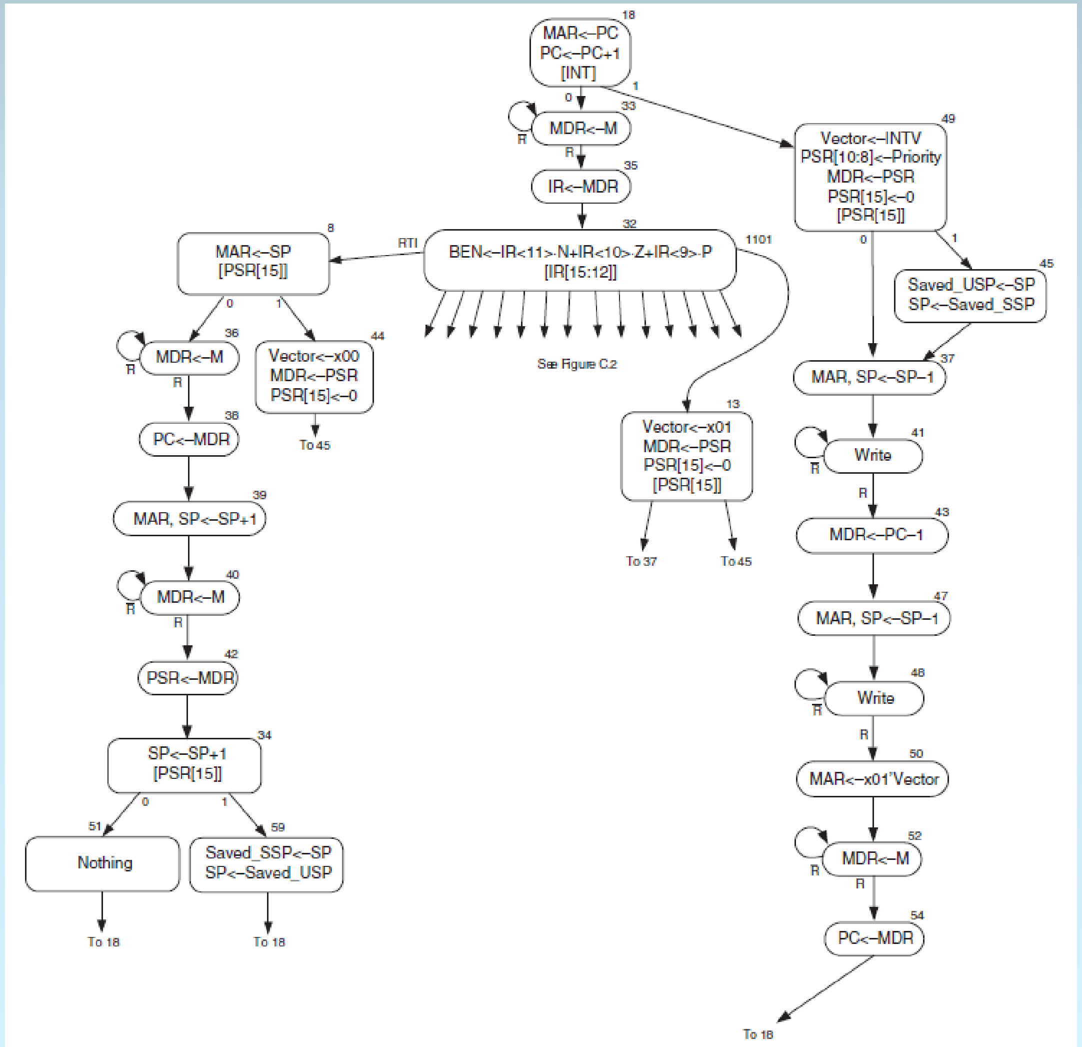


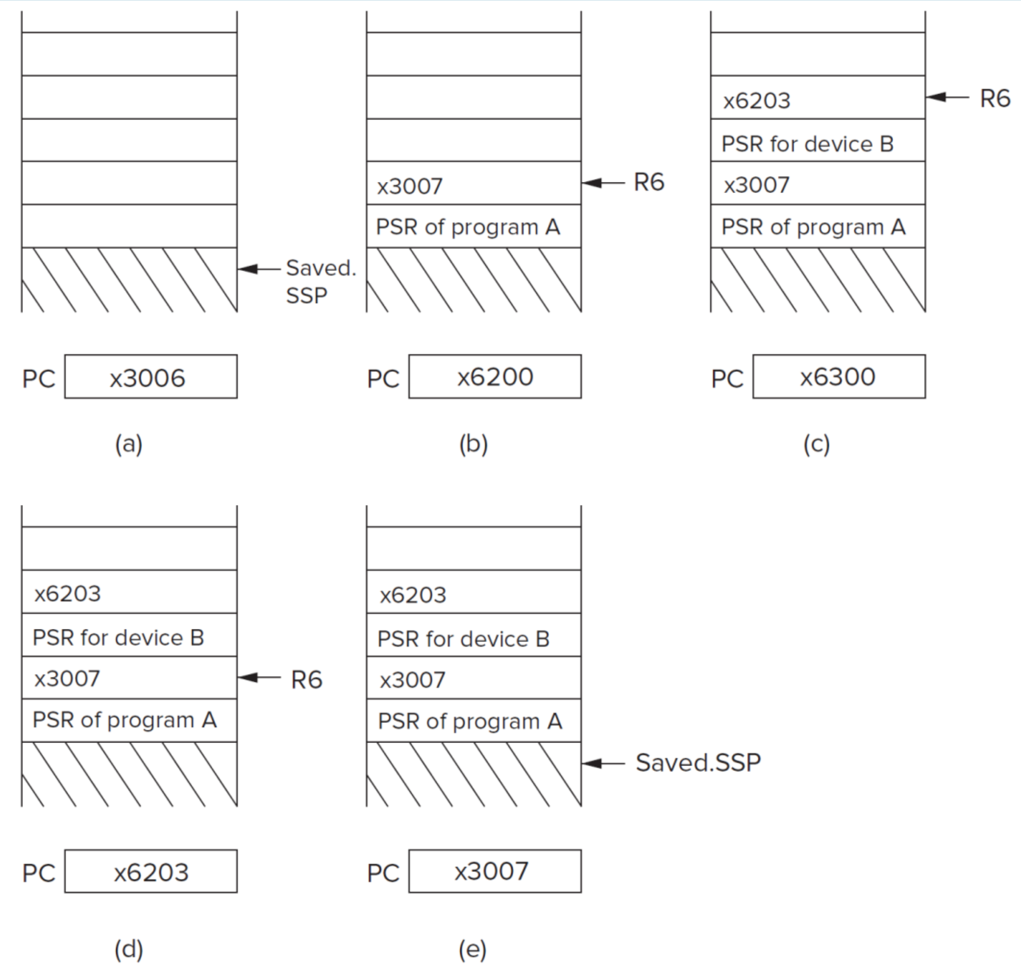
A.



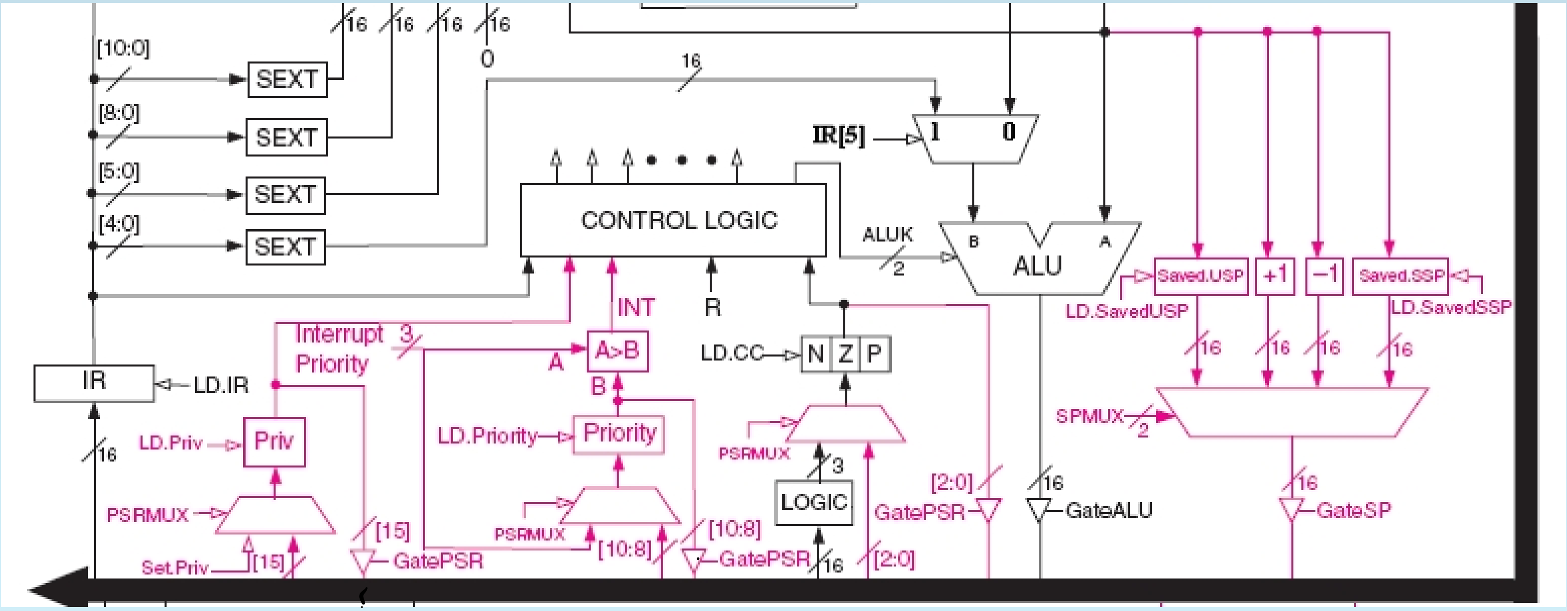
Content of Supervisory stack and PC during interrupt-driven I/O



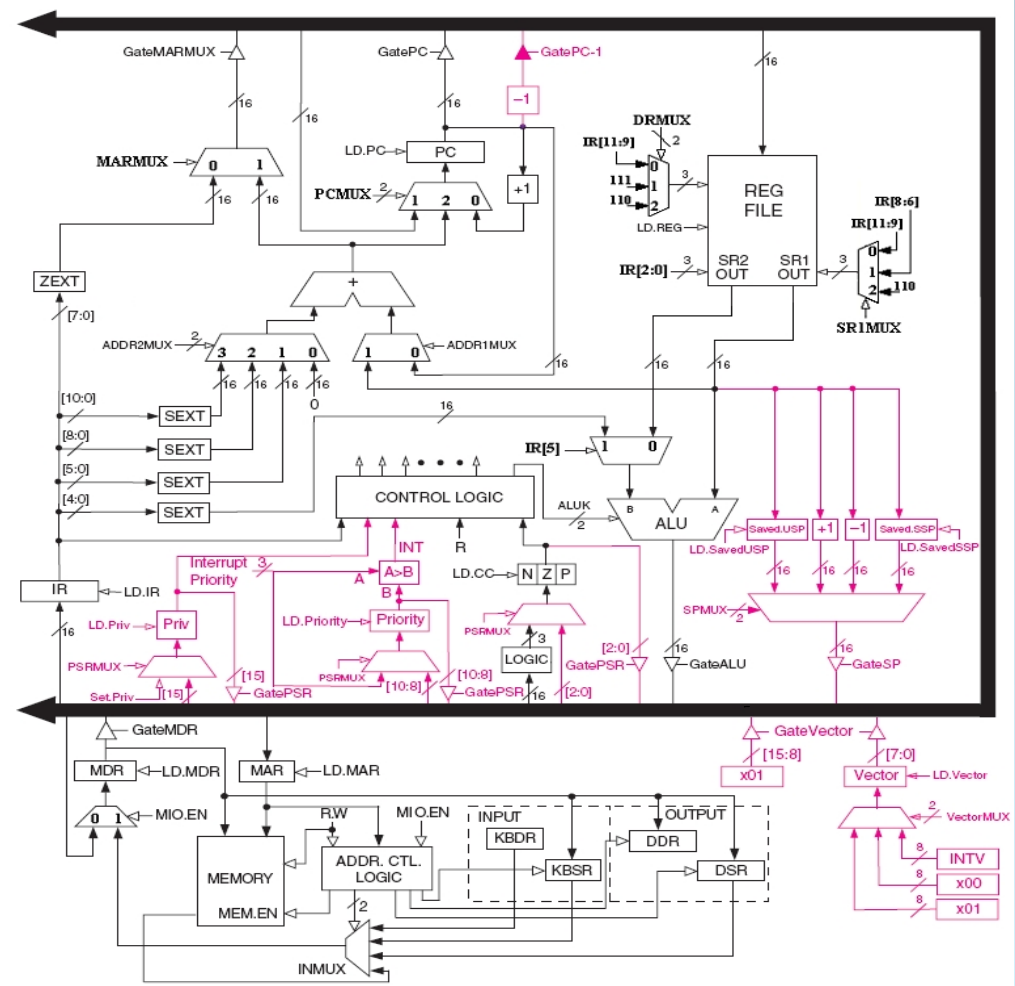


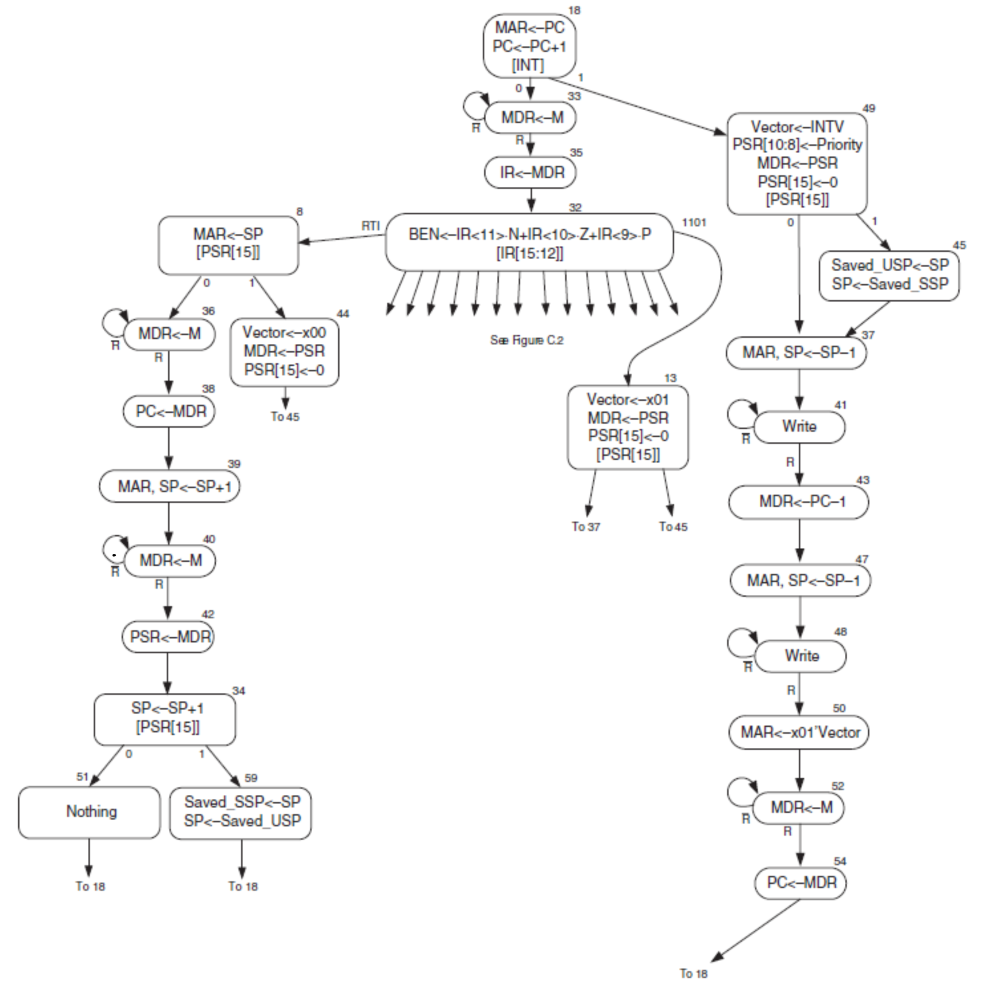


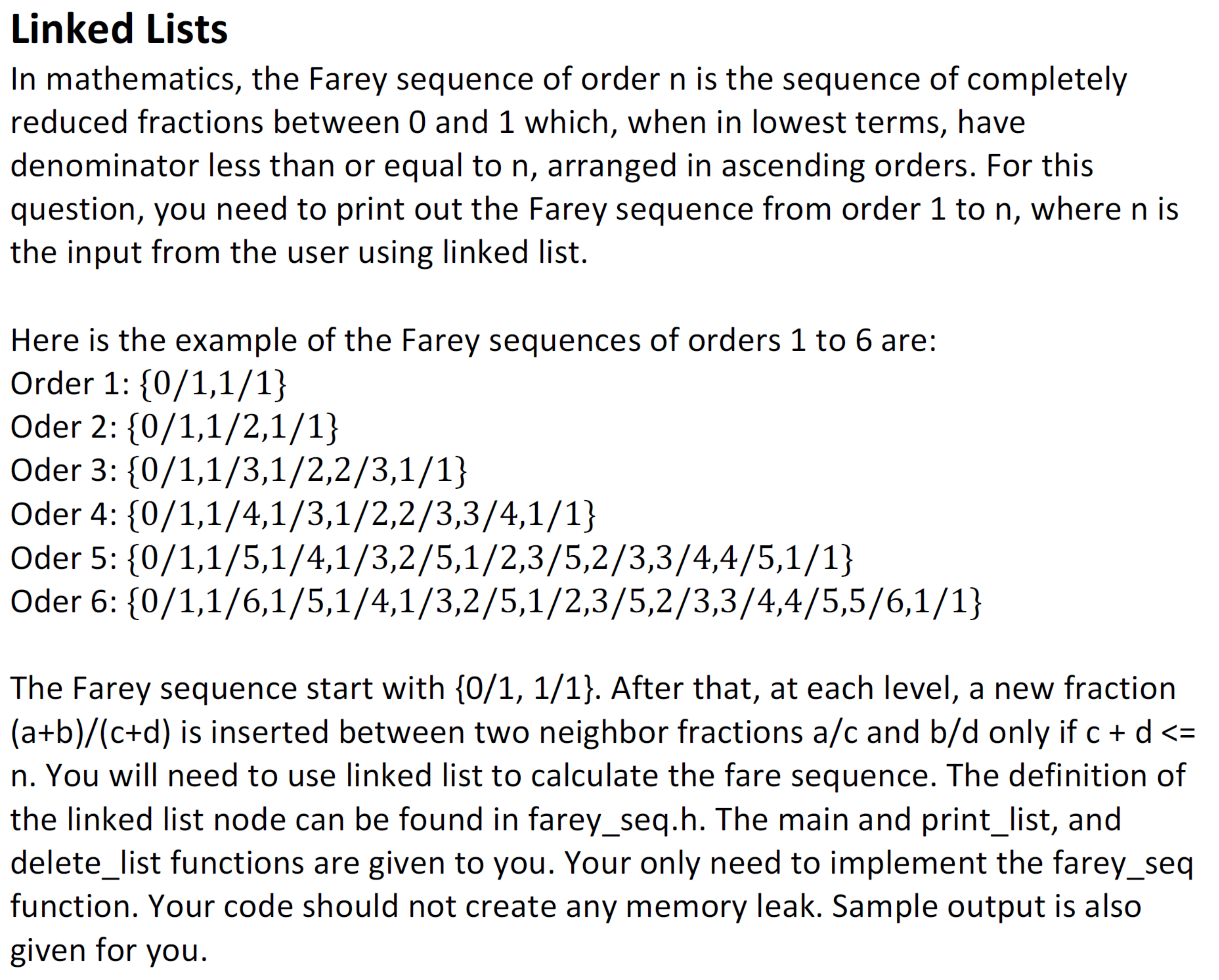
LC-3 Hardware to Support Interrupts



Extended LC-3 Datapath and FSM







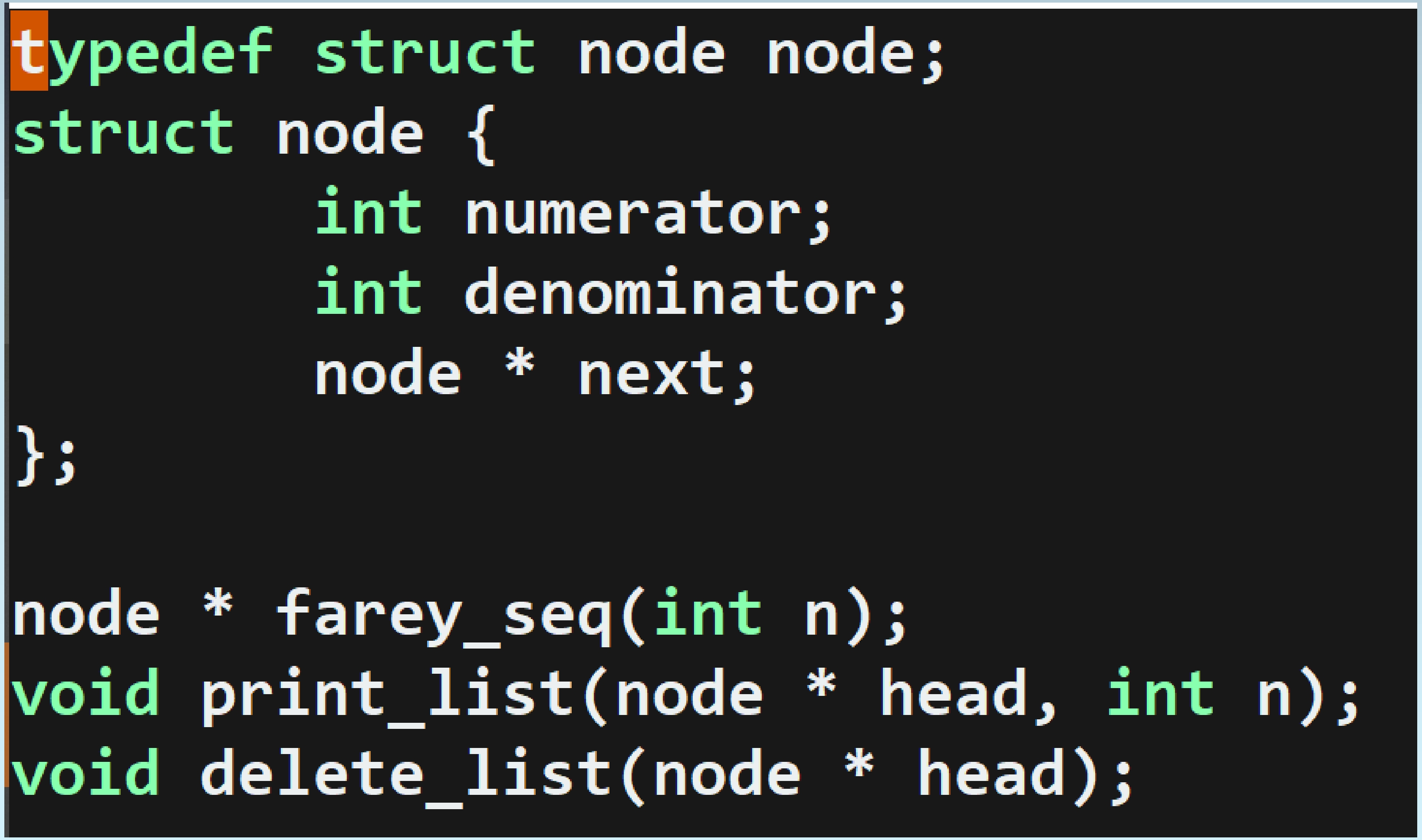
Linked Lists

In mathematics, the Farey sequence of order n is the sequence of completely reduced fractions between 0 and 1 which, when in lowest terms, have denominator less than or equal to n, arranged in ascending orders. For this question, you need to print out the Farey sequence from order 1 to n, where n is the input from the user using linked list.

Here is the example of the Farey sequences of orders 1 to 6 are:
Order 1: {0/1,1/1}
Oder 2: {0/1,1/2,1/1}
Oder 3: {0/1,1/3,1/2,2/3,1/1}
Oder 4: {0/1,1/4,1/3,1/2,2/3,3/4,1/1}
Oder 5: {0/1,1/5,1/4,1/3,2/5,1/2,3/5,2/3,3/4,4/5,1/1}
Oder 6: {0/1,1/6,1/5,1/4,1/3,2/5,1/2,3/5,2/3,3/4,4/5,5/6,1/1}

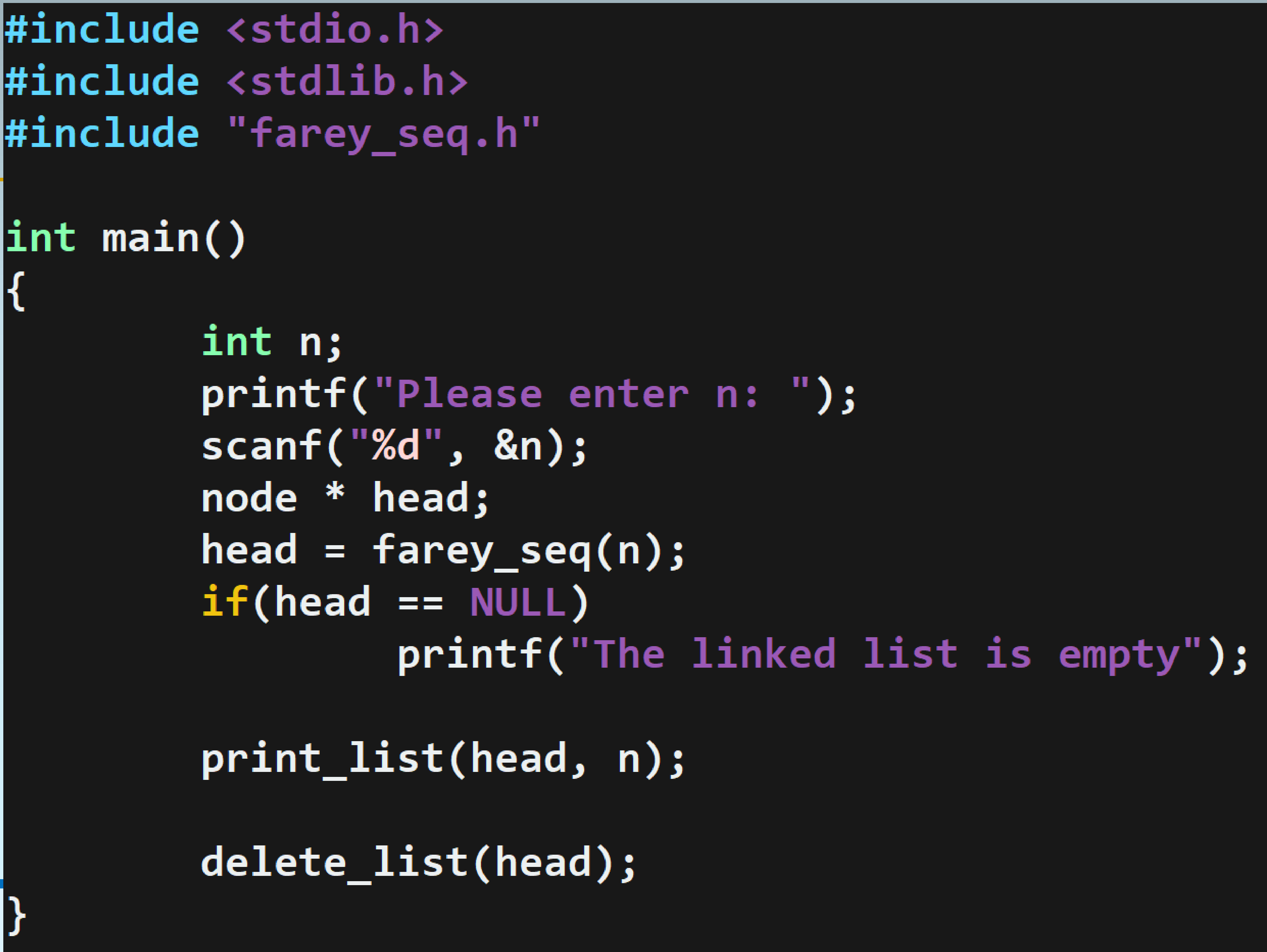
The Farey sequence start with {0/1, 1/1}. After that, at each level, a new fraction (a+b)/(c+d) is inserted between two neighbor fractions a/c and b/d only if c + d <= n. You will need to use linked list to calculate the fare sequence. The definition of the linked list node can be found in farey_seq.h. The main and print_list, and delete_list functions are given to you. Your only need to implement the farey_seq function. Your code should not create any memory leak. Sample output is also given for you.

farey_seq.h

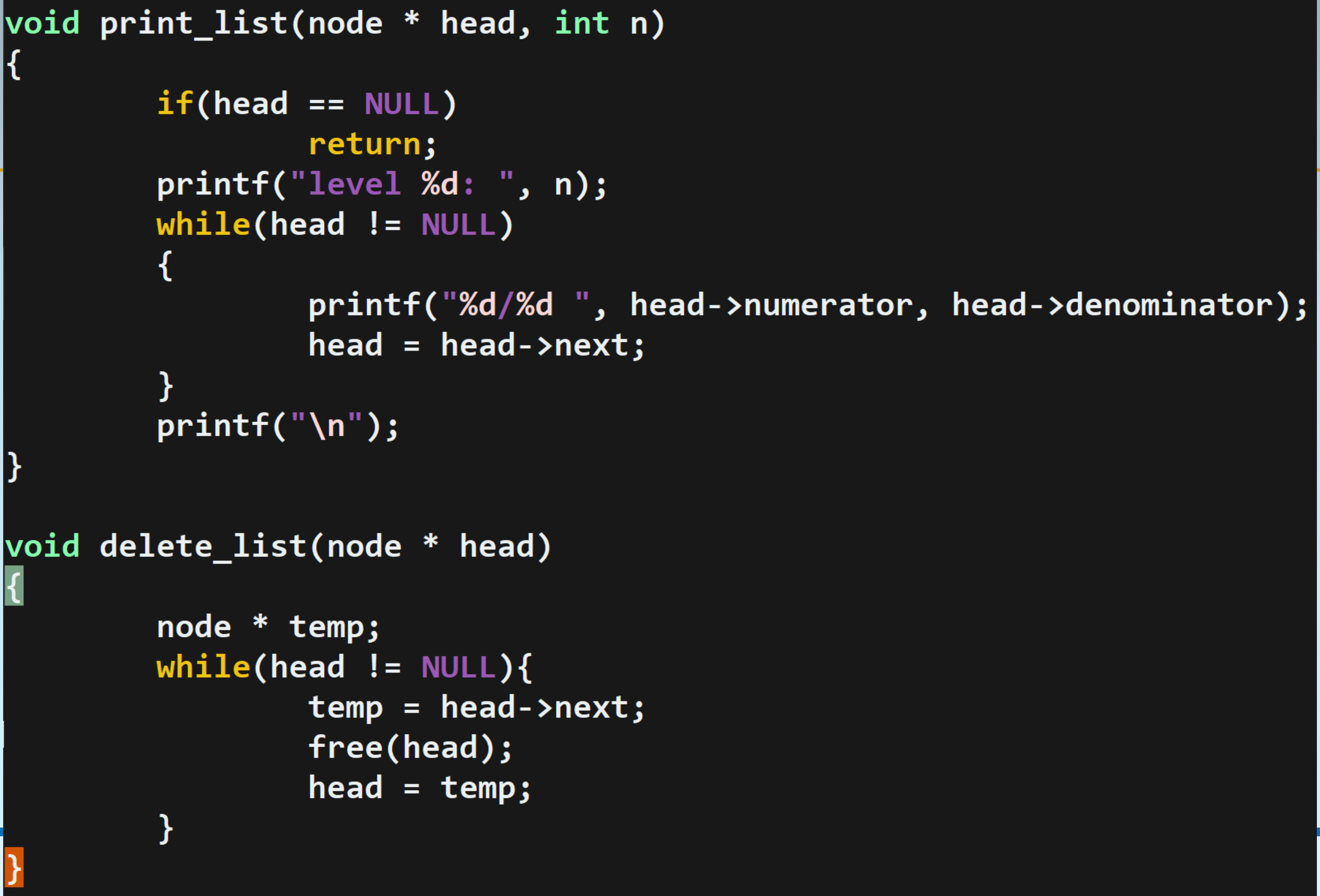


```c
typedef struct node node;
struct node {
        int numerator;
        int denominator;
        node * next;
};

node * farey_seq(int n);
void print_list(node * head, int n);
void delete_list(node * head);
```

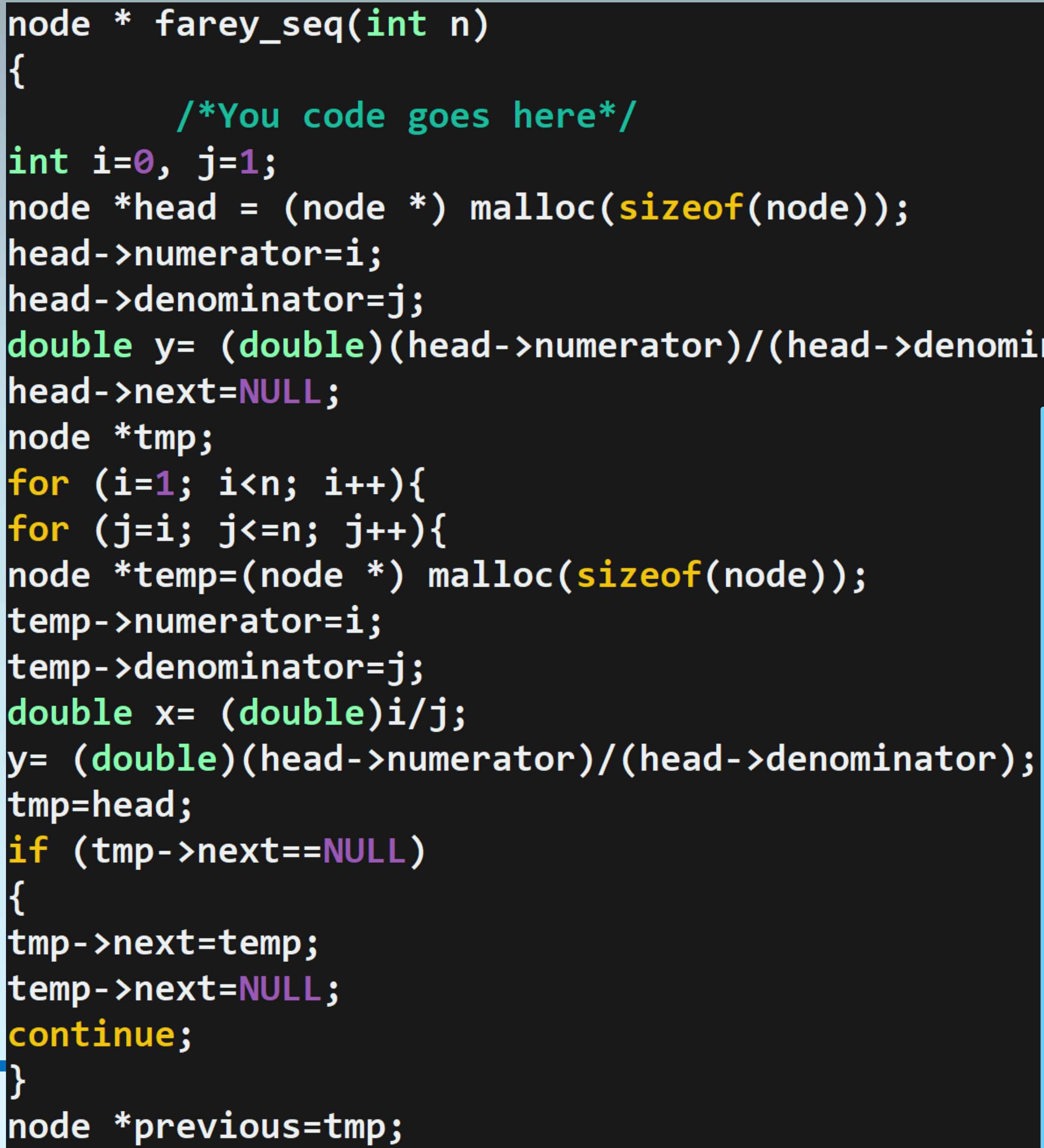
farey_seq.c



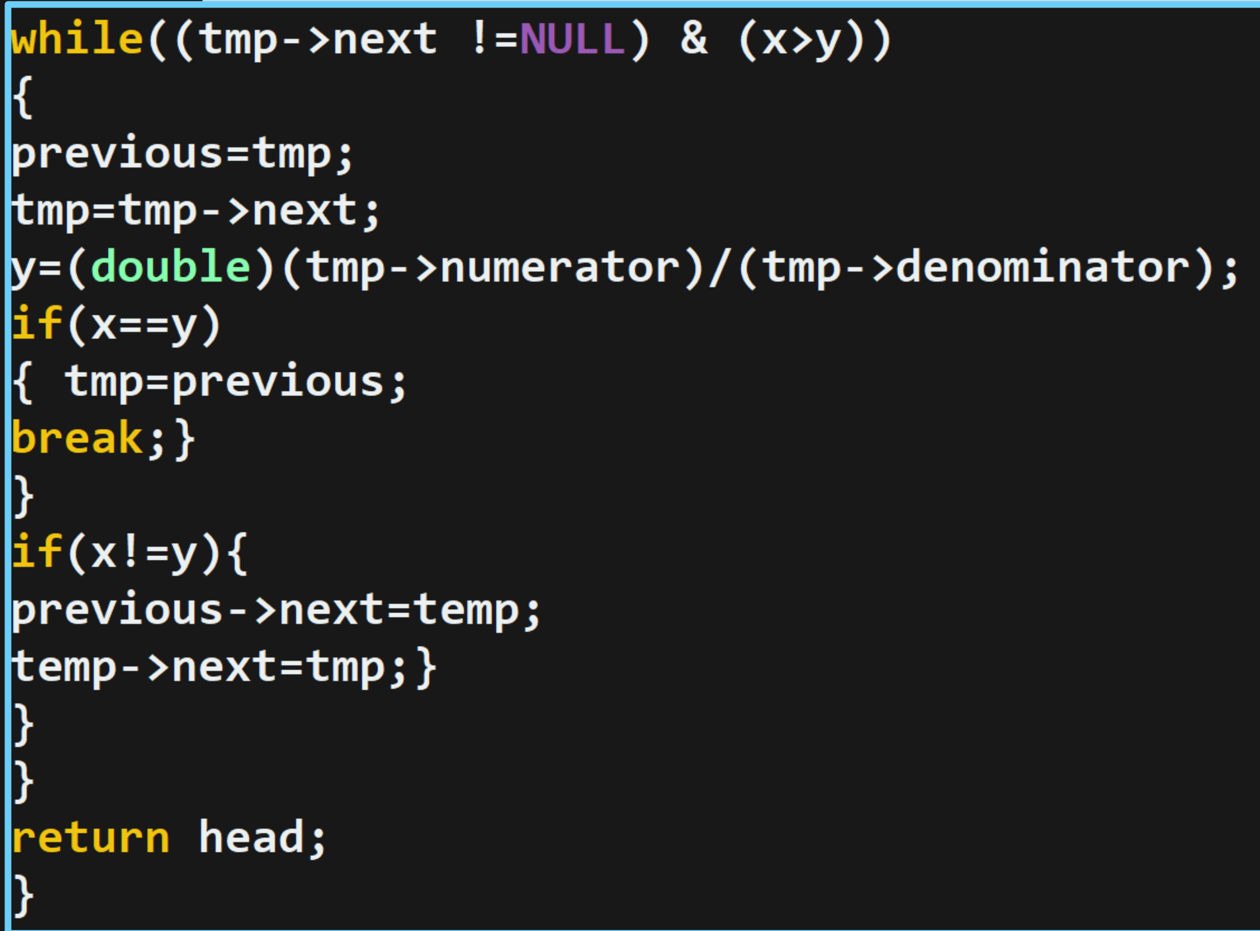
```c
#include <stdio.h>
#include <stdlib.h>
#include "farey_seq.h"

int main()
{
        int n;
        printf("Please enter n: ");
        scanf("%d", &n);
        node * head;
        head = farey_seq(n);
        if(head == NULL)
                printf("The linked list is empty");

        print_list(head, n);

        delete_list(head);
}
```



```
void print_list(node * head, int n)
{
        if(head == NULL)
                return;
        printf("level %d: ", n);
        while(head != NULL)
        {
                printf("%d/%d ", head->numerator, head->denominator);
                head = head->next;
        }
        printf("\n");
}

void delete_list(node * head)
{
        node * temp;
        while(head != NULL){
                temp = head->next;
                free(head);
                head = temp;
        }
}
```



```
node * farey_seq(int n)
{
        /*You code goes here*/
int i=0, j=1;
node *head = (node *) malloc(sizeof(node));
head->numerator=i;
head->denominator=j;
double y= (double)(head->numerator)/(head->denominator);
head->next=NULL;
node *tmp;
for (i=1; i<n; i++){
for (j=i; j<=n; j++){
node *temp=(node *) malloc(sizeof(node));
temp->numerator=i;
temp->denominator=j;
double x= (double)i/j;
y= (double)(head->numerator)/(head->denominator);
tmp=head;
if (tmp->next==NULL)
{
tmp->next=temp;
temp->next=NULL;
continue;
}
node *previous=tmp;
```



```
while((tmp->next !=NULL) & (x>y))
{
previous=tmp;
tmp=tmp->next;
y=(double)(tmp->numerator)/(tmp->denominator);
if(x==y)
{ tmp=previous;
break;}
}
if(x!=y){
previous->next=temp;
temp->next=tmp;}
}
}
return head;
}
```