Linked List

Project Requirements

Implement a program that manages a priority queue in a linked list. It is to be able to execute the following commands on the queue:

I - Insert data with priority

This command should allow the user to insert the ***character*** and its ***priority*** to be placed in the queue. If this operation cannot be performed due to *repeated priority* or *full buffer*, the program should indicate so and return to menu.

D - delete data

This command should allow the user to enter a ***character*** to be deleted from the queue. If there are more than one such ***character***, the one with the *highest priority* should be deleted. If there is no such ***character***, the program should indicate so and return to menu.

P -print queue

This command should prompt the program to print the elements of the queue in the decreasing order of their priority *(descending order)*. The order of print is *character*, *prev. address*, *next address*, and *priority* per line. Printed queue *stays* till return key is hit.

C -compact buffer

This command should prompt the program to perform compaction in the buffer, and print the content of the buffer after compaction.

The priority queue is to be implemented with a linked list, and is to be kept sorted at all times by decreasing order of priority.

Procedure:

a) Implement the program that has been described above.

b) The *priority* is to be a number between *00* and *99*, the *character* is a printable ASCII character, and *previous-* & *next-addresses* are each two bytes long.

c) A block of *120* bytes in RAM is to be set aside for the buffer.

d) Use a pointer to keep track of the available free space '*Empnode*' in the buffer. If pointer is out of range and buffer is not full, then program needs to perform compaction and print before insertion and notify the user.

e) When a compaction is called for, the free spaces may be collected *at the bottom* of the buffer.

f) Only ***two*** global variable pointers should be used. One variable to point at the head node '*Head*' and the other to point at available empty node '*Empnode*'. Any number may be used to represent the NULL pointer, but it should not conflict with the addresses in the buffer.

g) The element is guaranteed to be printable ASCII, so that clear byte can be used to represent '*free*' node in the buffer.

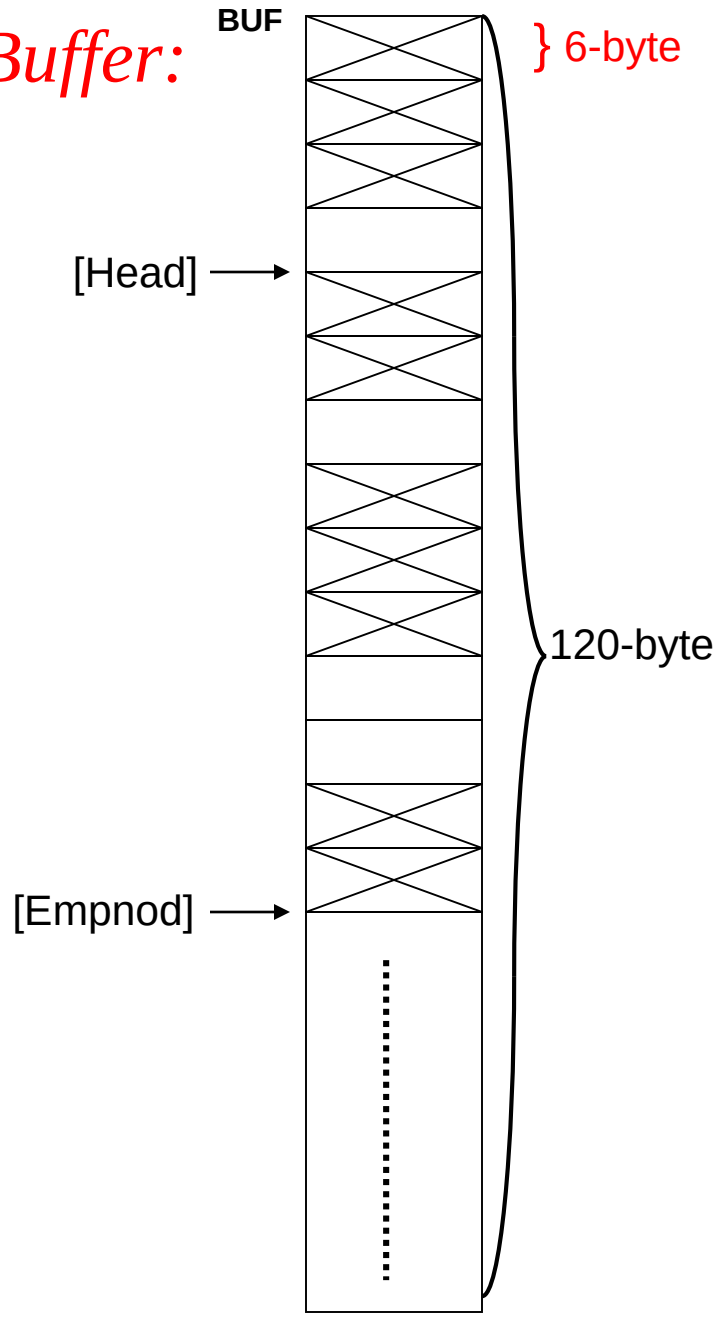
h) Assume that input is error-free.

Node:

D500	Previous Address$_H$
D501	Previous Address$_L$
D502	Next Address$_H$
D503	Next Address$_L$
D504	Priority
D505	ASCII Character

6-Bytes

Complete Buffer:



Parameters:

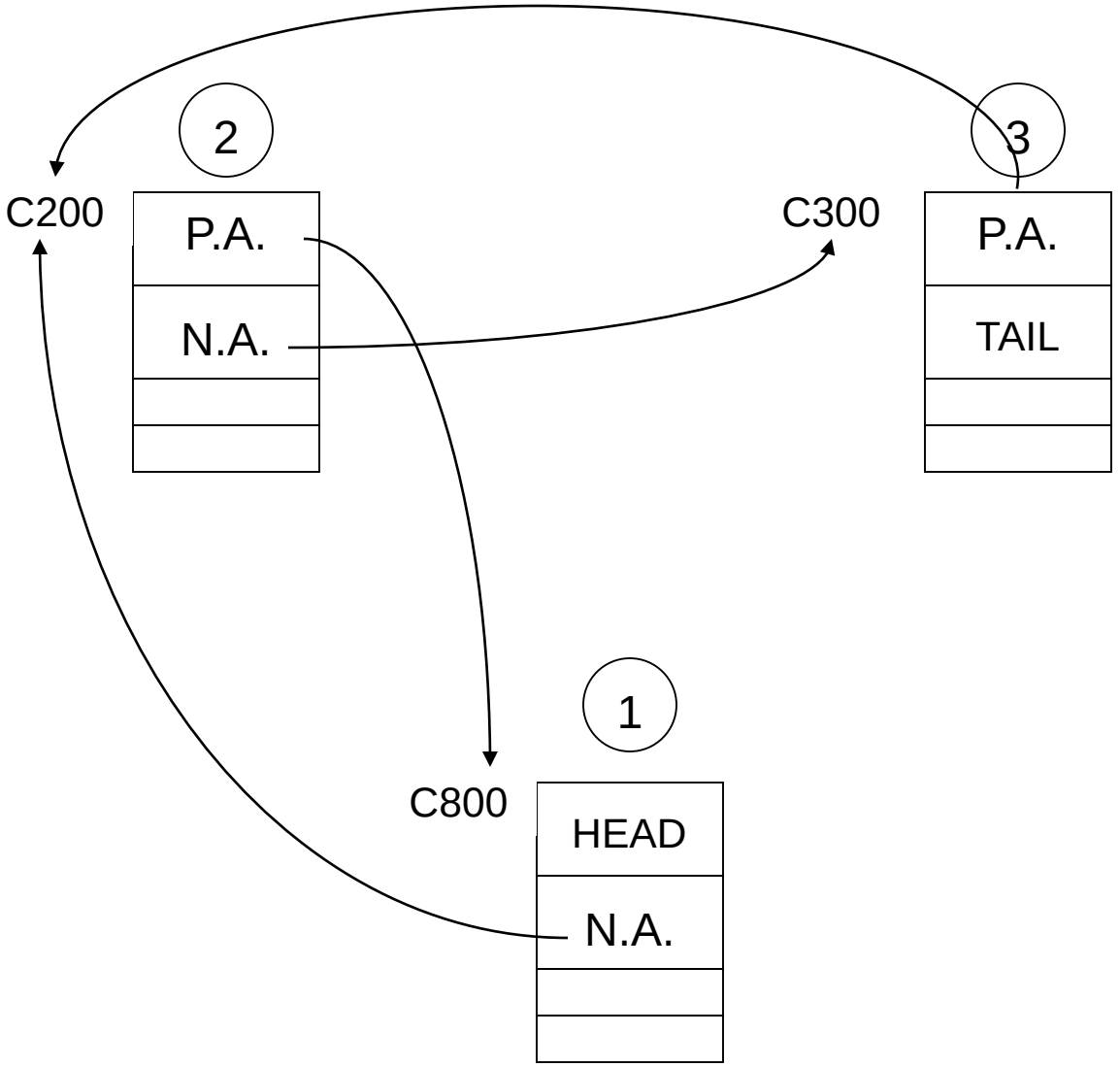
P.A. of Head Node: $0004
N.A. of Tail Node: $D601

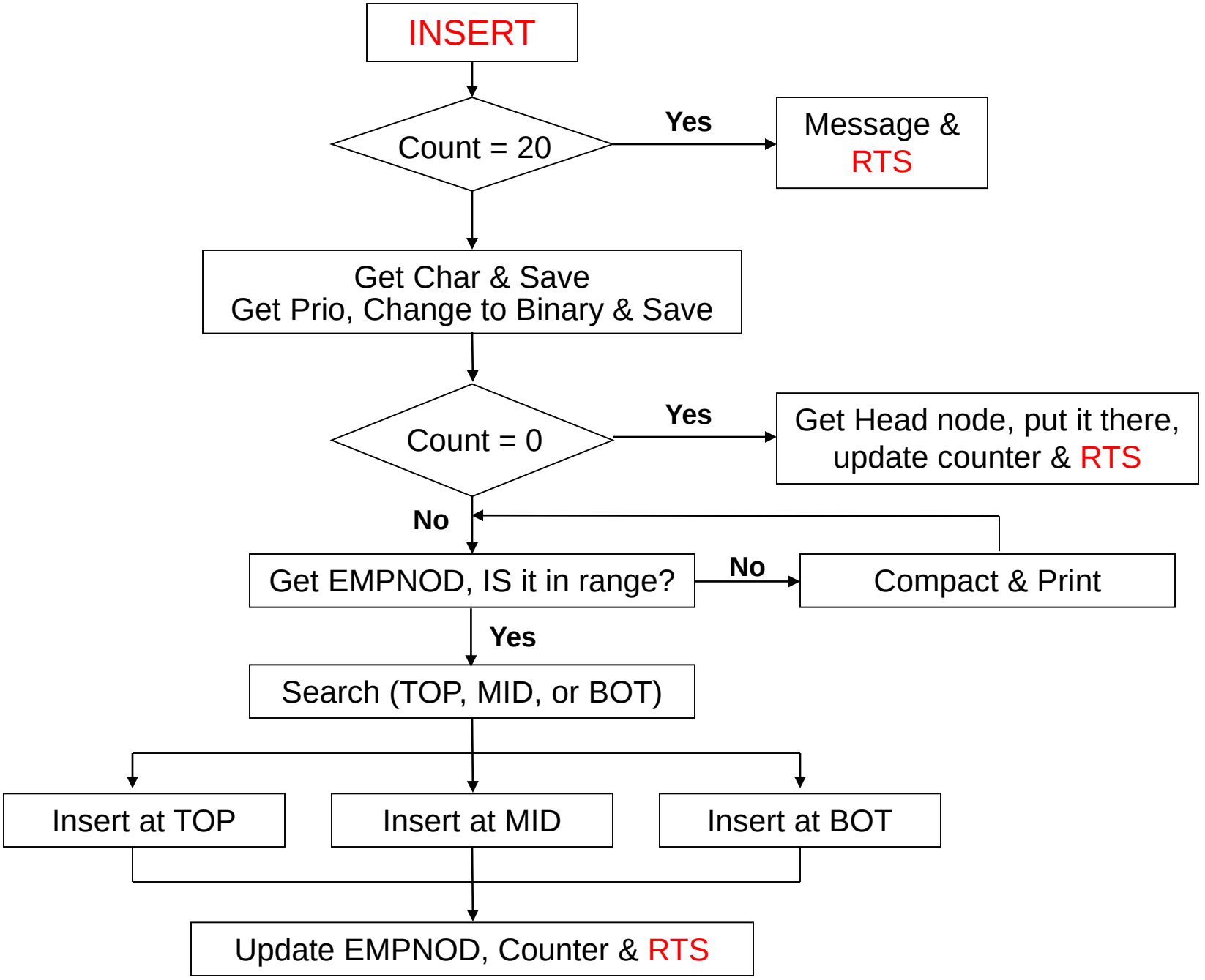
Pointers:

Head
Empnod

Variables:
Count (1byte), Prio (1byte), Char (1byte)

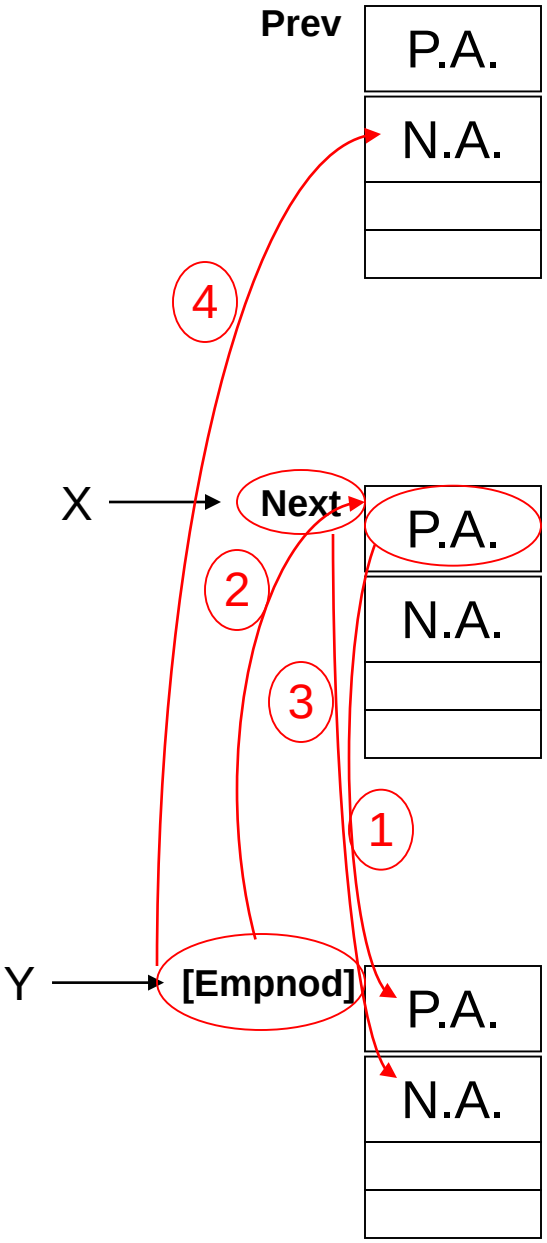
Linked List Concept:

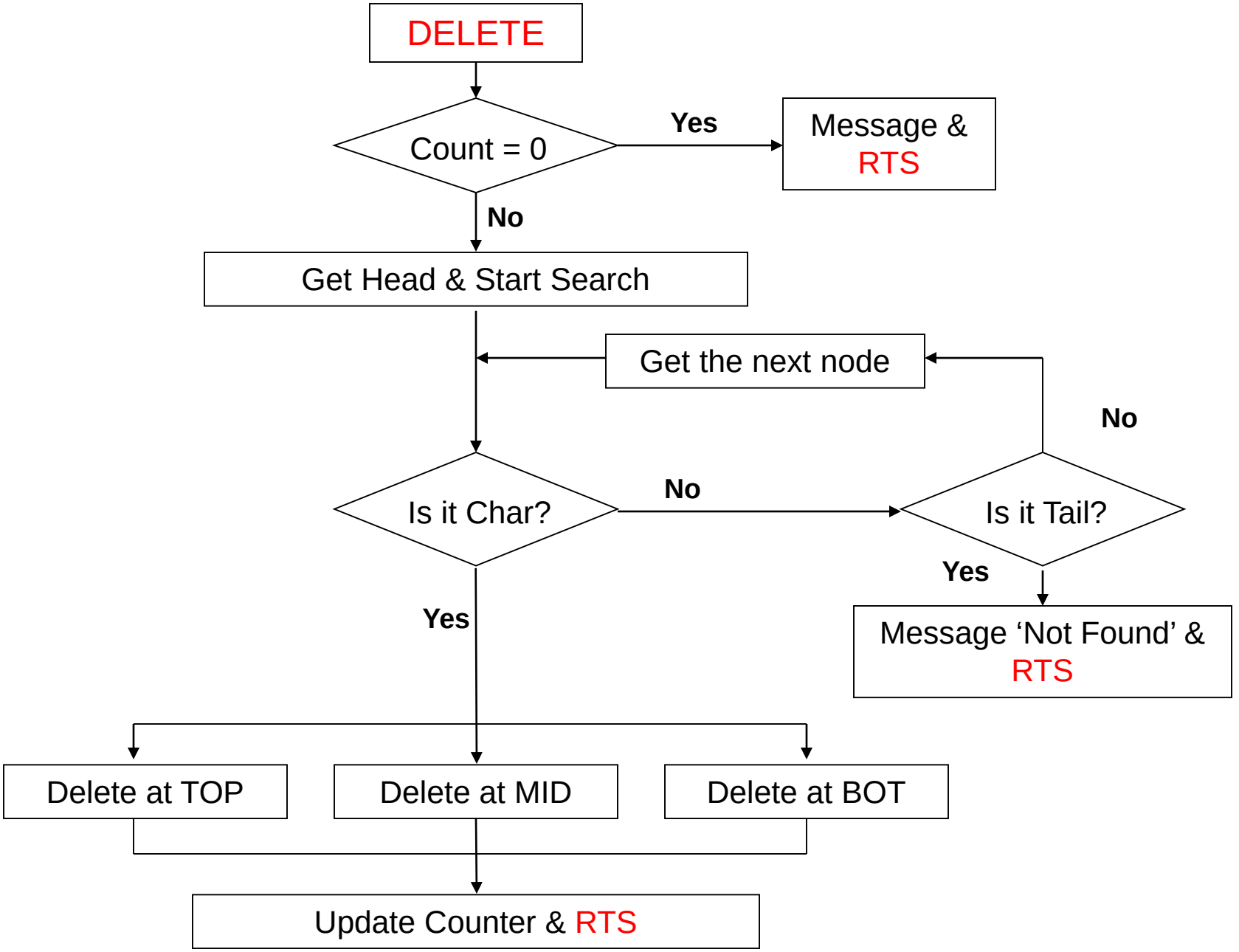



INSERT
Count = 20
Yes
Message & RTS
Get Char & Save
Get Prio, Change to Binary & Save
Count = 0
Yes
Get Head node, put it there, update counter & RTS
No
Get EMPNOD, IS it in range?
No
Compact & Print
Yes
Search (TOP, MID, or BOT)
Insert at TOP
Insert at MID
Insert at BOT
Update EMPNOD, Counter & RTS

Insert in the Middle:

```
LDD     0,X   } 1
STD     0,Y   }
STY     0,X     2
STX     2,Y     3
XGDX            }
STY     2,X   } 4
MOVB    prio,4,Y  }
MOVB    char,5,Y  } 5
```



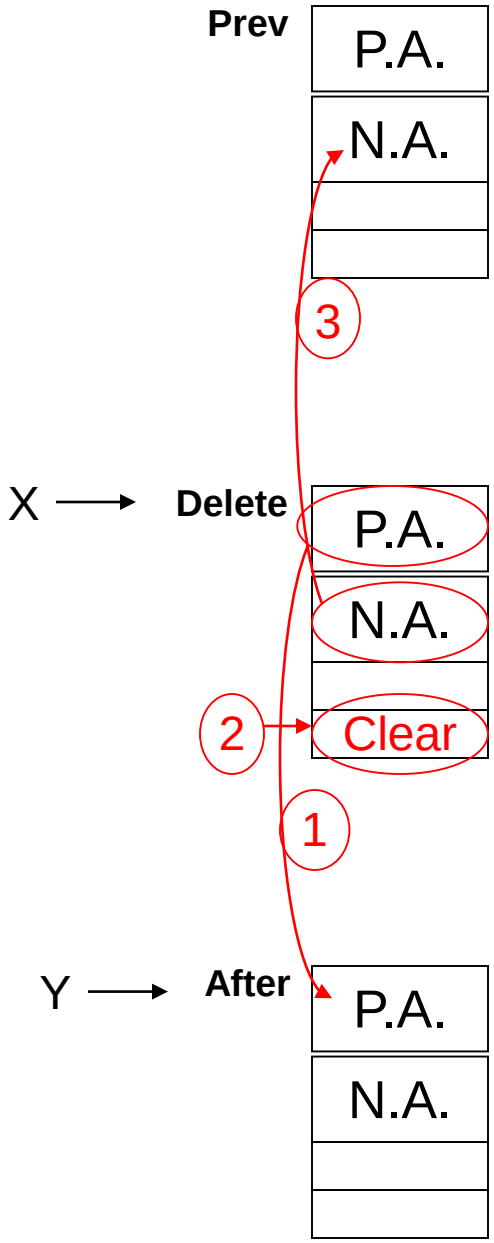

DELETE
Count = 0
Yes
Message & RTS
No
Get Head & Start Search
Get the next node
No
Is it Char?
No
Is it Tail?
Yes
Yes
Message 'Not Found' & RTS
Delete at TOP
Delete at MID
Delete at BOT
Update Counter & RTS

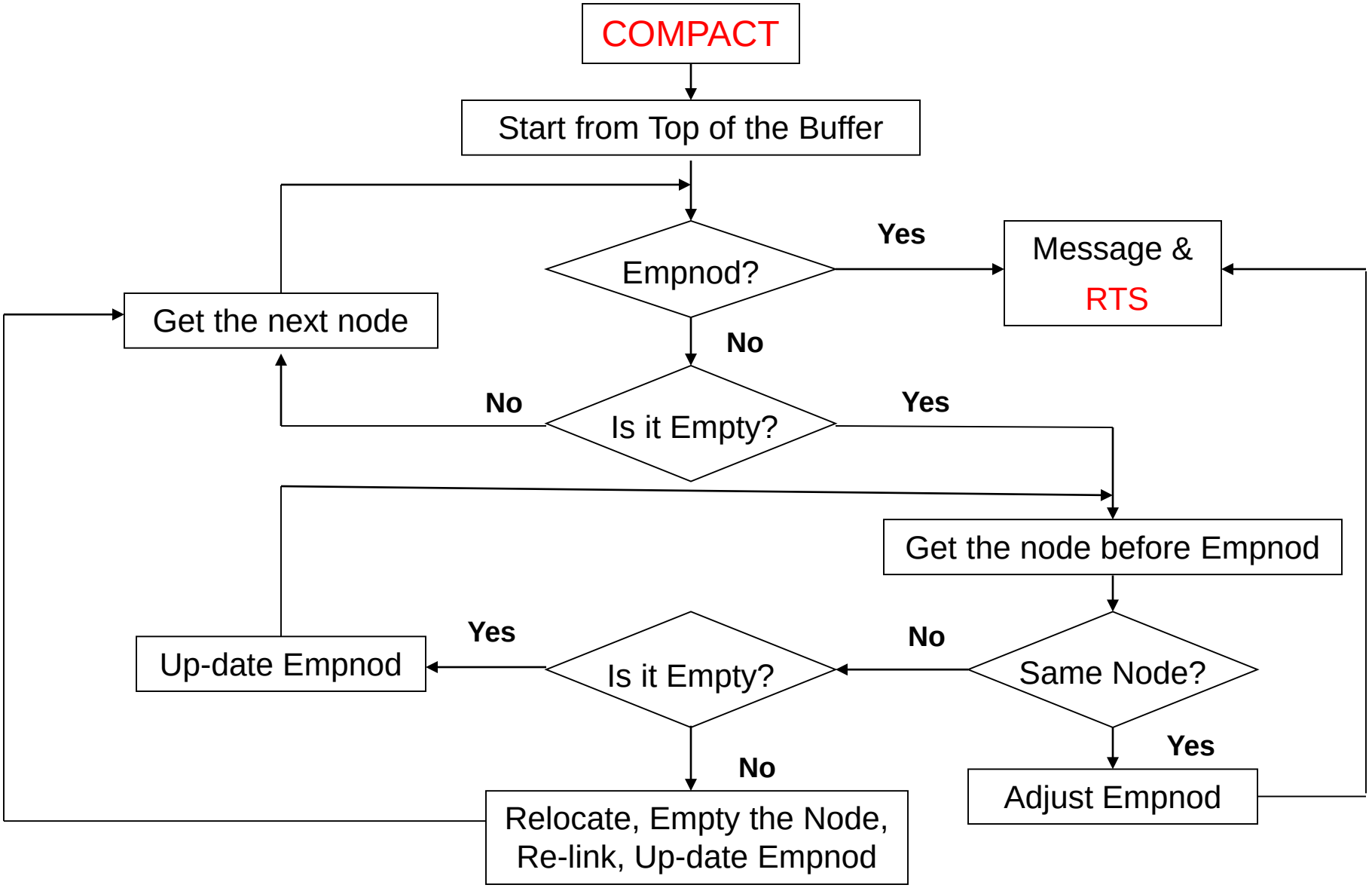
Delete in the Middle:

```
LDY     2,X        } 1
MOVW    0,X,0,Y    }

CLR     5,X          2

LDX     0,X        } 3
STY     2,X        }
```




COMPACT
Start from Top of the Buffer
Empnod?
Yes
Message &
RTS
No
Get the next node
No
Is it Empty?
Yes
Get the node before Empnod
Yes
Up-date Empnod
Is it Empty?
No
Same Node?
Yes
Adjust Empnod
No
Relocate, Empty the Node,
Re-link, Up-date Empnod

Compact:

```
MOVW   0,Y,0,X
MOVW   2,Y,2,X
MOVW   4,Y,4,X
```

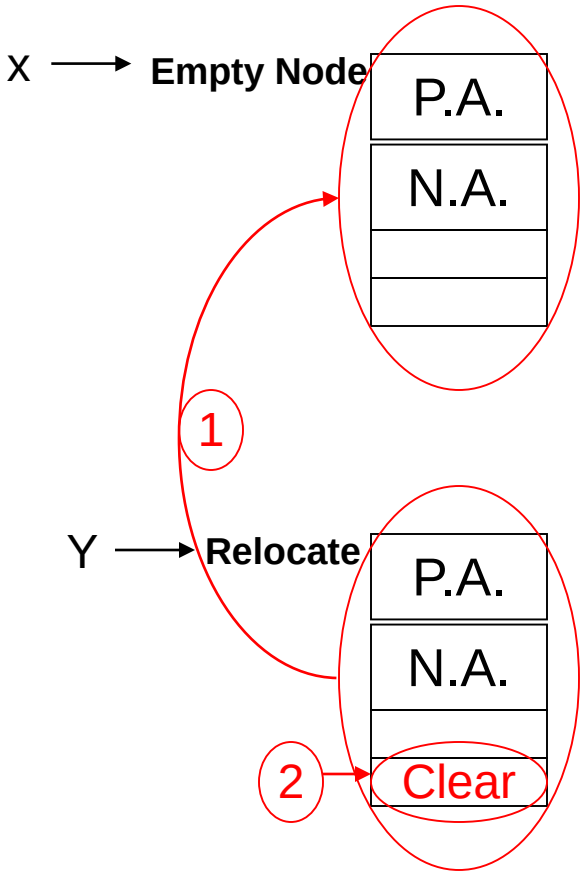
1 Relocate

```
CLR    5,Y
STY    Empnod
```

2 Empty & Update

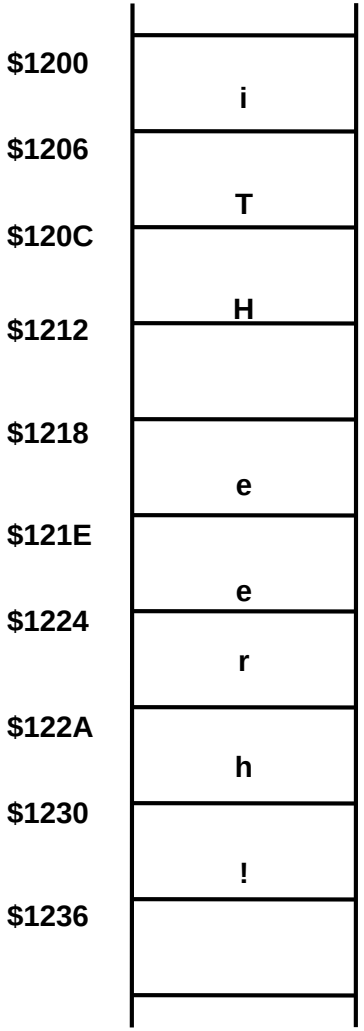
```
LDY    0,X
STX    2,Y
LDY    2,X
STX    0,Y
```

3 Re-link



Printing:

Char	Prev	Next	Priority
H	0004	1200	95
i	120C	1212	85
	1200	1206	80
T	1212	122A	70
h	1206	1218	66
e	122A	1224	60
r	1218	121E	52
e	1224	1230	44
!	121E	D601	32



Printing:

```
        ORG     $1000

PRINT   LDD     #prmsg
        LDX     printf
        JSR     0,X
        LDAB    Count
        BNE     norm
        RTS

norm    LDY     Head
redo    PSHY
        LDAB    5,Y
        CLRA
        LDX     putchar
        JSR     0,X
        PULY
        PSHY
        LDAB    4,Y
        CLRA
        PSHD
        LDD     2,Y
        PSHD
        LDD     0,Y
        PSHD
        LDD     #nprmsg
        LDX     printf
        JSR     0,X
        LEAS    6,SP
        PULY
        LDY     2,Y
        CPY     #$D601
        BNE     redo
        LDD     #contmsg
        LDX     printf
        JSR     0,X
        LDX     getchar
        JSR     0,X
        RTS
```

```
            ORG     $2000
Buff        DB      $20,$0c,$20,$18,80,$65
            DB      $20,$18,$20,$12,60,$6C
            DB      $10,$00,$20,$00,90,$48
            DB      $20,$06,$D6,$01,50,$6F
            DB      $20,$00,$20,$06,70,$6C
Count       DB      5
Head        DW      $200C
printf      EQU     $EE88
putchar     EQU     $EE86
getchar     EQU     $EE84
prmsg       DB      $0D,$0D
            FCC     'Char    P.A.    N.A.   Prio'
            DB      $0D
            FCC     '                                        '
            DB      $0D,$0A,0
nprmsg      FCC     '       %X    %X    %u '
            DB      $0D,$0A,0
contmsg     FCC      ' Hit any key to continue: ‘
            DB      $0D,$0A,0
```