Assignment #7

Problem 7.1: *positioning algorithms* (1+1+1+1 = 4 points)

A system using segmentation has free blocks of the following sizes:

```
17 KiB,  8 KiB, 10 KiB, 21 KiB, 12 KiB, 13 KiB
```

The blocks are ordered as shown above and the last memory allocation took place right before the first block. The system receives the following sequence of memory requests (in the order shown below):

```
11 KiB,  9 KiB,  7 KiB, 16 KiB
```

If a free block is allocated that is bigger than the request, then a new small block is left over and positioned at the same place where the bigger block was positioned in the list of free blocks.

a) Show how the list of free blocks changes after each allocation using the best-fit algorithm.

	17 KiB,	8 KiB,	10 KiB,	21 KiB,	12 KiB,	13 KiB
11 KiB:	17 KiB	8 KiB	10 KiB	21 KiB	1 KiB	13 KiB
9 KiB:	17 KiB	8 KiB	1 KiB	21 KiB	1 KiB	13 KiB
7 KiB:	17 KiB	1 KiB	1 KiB	21 KiB	1 KiB	13 KiB
16 KiB:	1 KiB	1 KiB	1 KiB	21 KiB	1 KiB	13 KiB

b) Show how the list of free blocks changes after each allocation using the worst-fit algorithm.

	17 KiB,	8 KiB,	10 KiB,	21 KiB,	12 KiB,	13 KiB
11 KiB:	17 KiB	8 KiB	10 KiB	10 KiB	12 KiB	13 KiB
9 KiB:	8 KiB	8 KiB	10 KiB	10 KiB	12 KiB	13 KiB
7 KiB:	8 KiB	8 KiB	10 KiB	10 KiB	12 KiB	6 KiB
16 KiB:			Memory	Full!		

c) Show how the list of free blocks changes after each allocation using the first-fit algorithm.

	17 KiB,	8 KiB,	10 KiB,	21 KiB,	12 KiB,	13 KiB
11 KiB:	6 KiB	8 KiB	10 KiB	21 KiB	12 KiB	13 KiB
9 KiB:	6 KiB	8 KiB	1 KiB	21 KiB	12 KiB	13 KiB
7 KiB:	6 KiB	1 KiB	1 KiB	21 KiB	12 KiB	13 KiB
16 KiB:	6 KiB	1 KiB	1 KiB	5 KiB	12 KiB	13 KiB

d) Show how the list of free blocks changes after each allocation using the next-fit algorithm.

	17 KiB,	8 KiB,	10 KiB,	21 KiB,	12 KiB,	13 KiB
11 KiB:	6 KiB	8 KiB	10 KiB	21 KiB	12 KiB	13 KiB
9 KiB:	6 KiB	8 KiB	1 KiB	21 KiB	12 KiB	13 KiB
7 KiB:	6 KiB	8 KiB	1 KiB	14 KiB	12 KiB	13 KiB
16 KiB:			Memory	Full!		

Problem 7.2: *buddy system* (1+1+1 = 3 points)

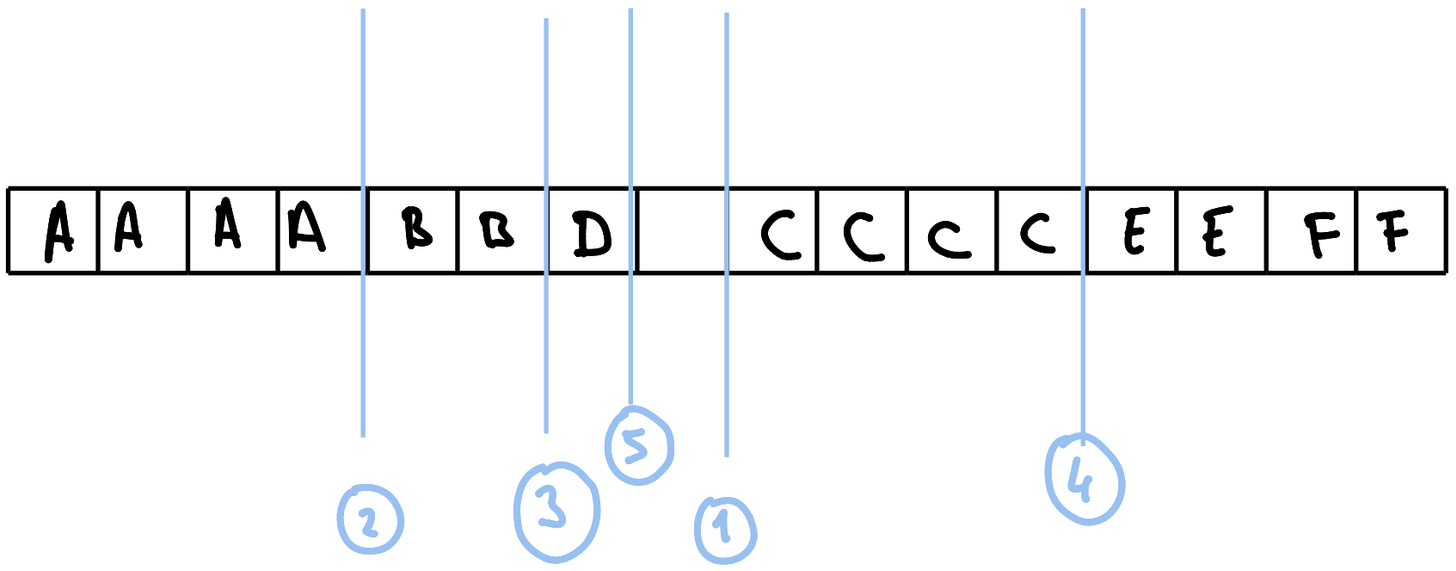
A system is using a buddy system memory allocator and it has initially 512 KiB of free memory available for memory allocation. The processes request memory in the following sequence:

```
A: +113 KiB,  B: +56 KiB,  C: +82 KiB,  D: +30 KiB,  E: +42 KiB,  F: +48 KiB
```

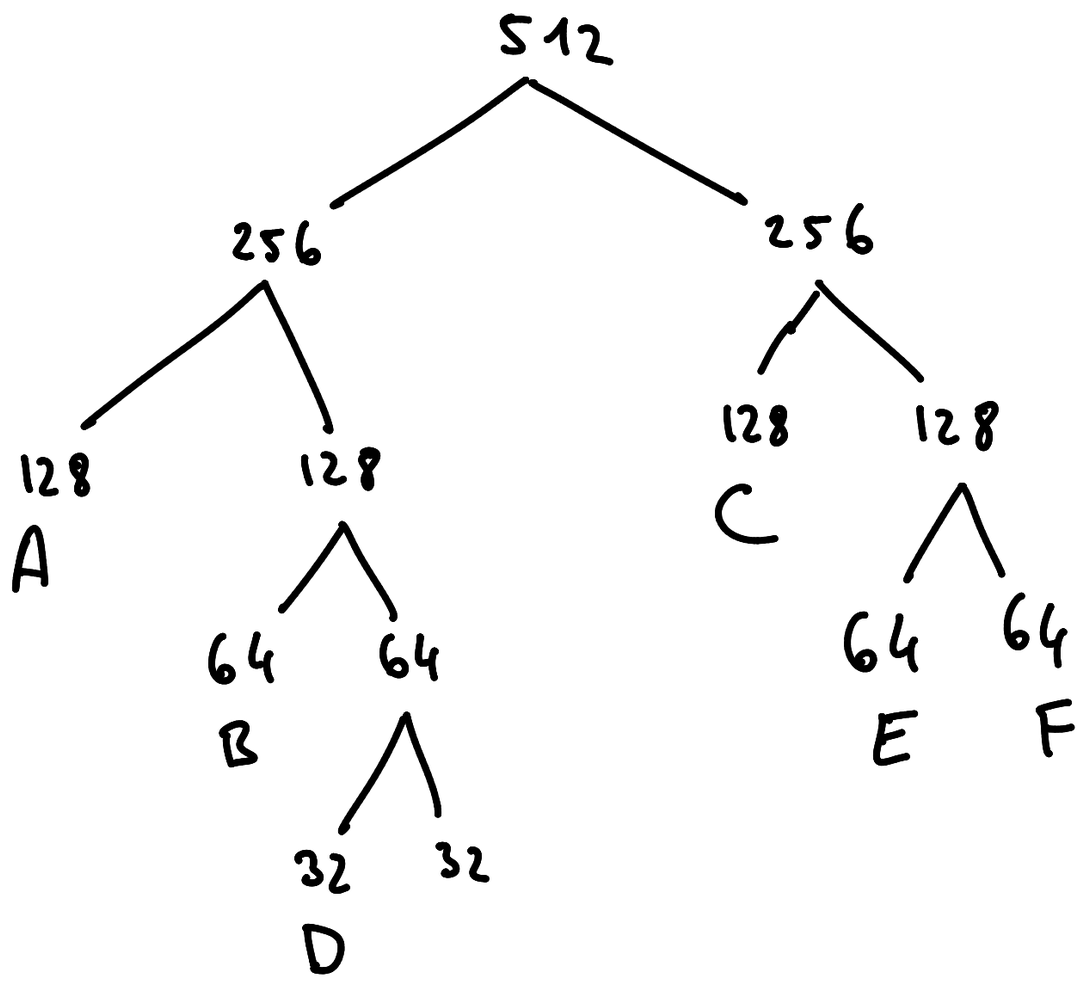
a) 128 KiB, 64 KiB, 128 KiB, 32 KiB, 64 KiB, 64 KiB

a) Draw a diagram showing how the allocations are placed in the 512 KiB of free memory. Draw also the associated binary tree.

Note! Each square is 32 KiB.



In KiB



b) Calculate the overall internal fragmentation. What is the largest chunk of memory that can still be allocated?

In KiB:

A ⇒ 128 − 113 = 15

B ⇒ 64 − 56 = 8

C ⇒ 128 − 82 = 46

D ⇒ 32 − 30 = 2

E ⇒ 64 − 42 = 22

F ⇒ 64 − 48 = 16

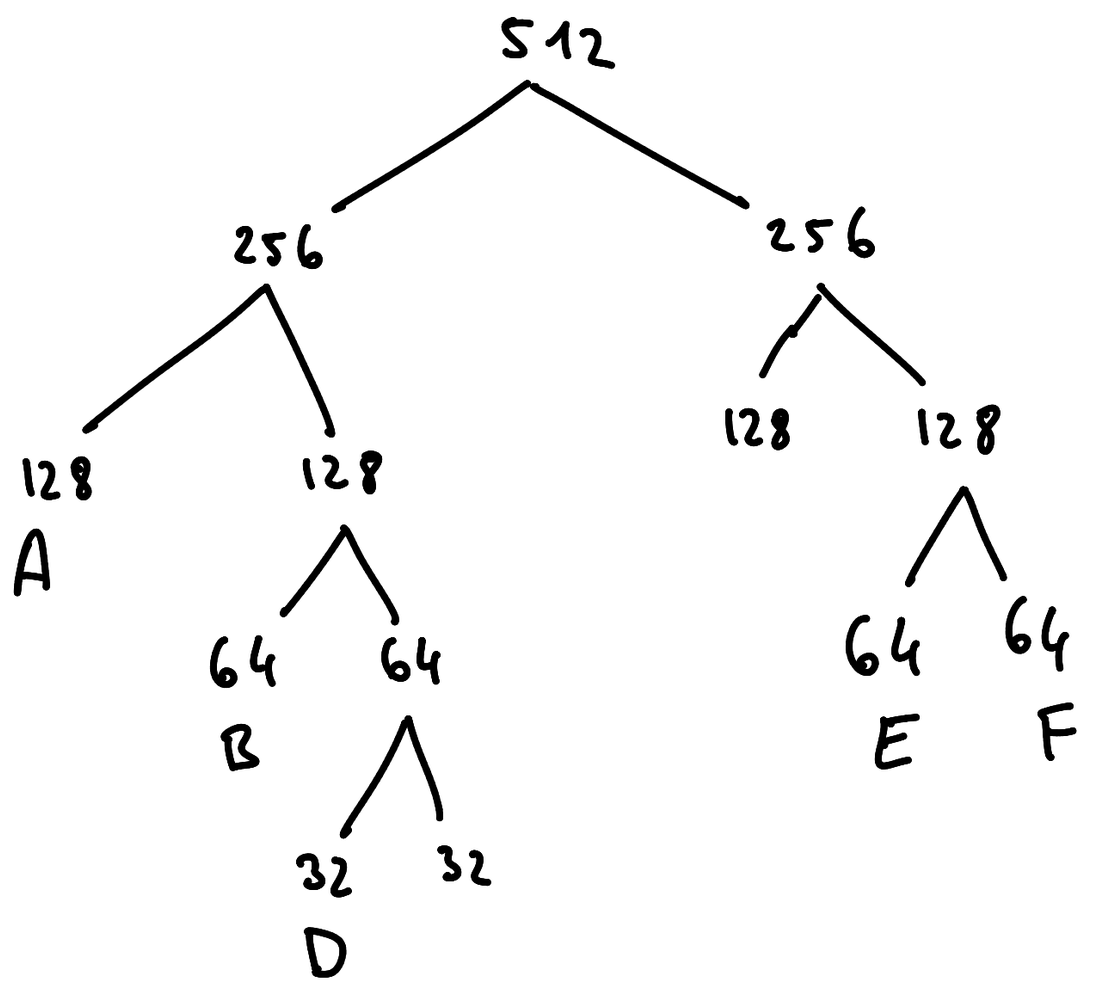
∴ Total = 109

The largest available chunk is 32 KiB. (Chunk after D).

c) Suppose process C returns its allocation, can a subsequent allocation G: 132 KiB be accomodated? Explain why or why not.

After C freed:

In KiB



We cannot combine the 32 KiB chunk with the 128 KiB chunk, because they are on separate branches.

⇒ largest available chunk = 128 KiB.

⇒ G will not fit.

Problem 7.3: *page replacement algorithms* (1+1+1 = 3 points)

An operating system uses demand paging to implement virtual memory. A user-space process uses four pages, numbered 1 to 4. The pages are accessed in the following order (reference string): 1, 4, 2, 3, 4, 4, 1, 3, 2, 1. None of the pages used by the user-space process are resident when the process starts and the frames allocated to the process are unused. Fill out the following tables by writing page numbers into the cells. Each number indicates which page a frame holds after accessing the page indicated by the reference string. Write down the number of page faults for each scenario.

↳ PF

a) Show how the pages are mapped to two frames using the First-In-First-Out (FIFO) page replacement algorithm.

reference string	1	4	2	3	4	4	1	3	2	1
frame 0	1	1	2	2	4	4	4	3	3	1
frame 1		4	4	3	3	3	1	1	2	2
	X	X	X	X	X		X	X	X	X

PF = 9

Show how the pages are mapped to three frames using the First-In-First-Out (FIFO) page replacement algorithm.

reference string	1	4	2	3	4	4	1	3	2	1
frame 0	1	1	1	3	3	3	3	3	3	3
frame 1		4	4	4	4	4	1	1	1	1
frame 2			2	2	2	2	2	2	2	2
	X	X	X	X			X			

PF = 5

b) Show how the pages are mapped to two frames using Belady's Optimal (BO) page replacement algorithm.

Note: LU ⇒ longest unused

LU: — — 1 2 — — 4 — 3

reference string	1	4	2	3	4	4	1	3	2	1
frame 0	1	1	2	3	3	3	3	3	2	2
frame 1		4	4	4	4	4	1	1	1	1
	X	X	X	X			X		X	

PF = 6

Show how the pages are mapped to three frames using Belady's Optimal (BO) page replacement algorithm.

LU: — — — 2 4

reference string	1	4	2	3	4	4	1	3	2	1
frame 0	1	1	1	1	1	1	1	1	1	1
frame 1		4	4	4	4	4	4	4	2	2
frame 2			2	3	3	3	3	3	3	3
	X	X	X	X					X	

PF = 5

c) Show how the pages are mapped to two frames using the Least Recently Used (LRU) page replacement algorithm.

QUEUE : {1} {4,1} {2,4} {3,2} {4,3} {4,3} {1,4} {3,1} {2,3} {1,2}

reference string	1	4	2	3	4	4	1	3	2	1
frame 0	1	1	2	2	4	4	4	3	3	1
frame 1		4	4	3	3	3	1	1	2	2
	X	X	X	X	X		X	X	X	X

PF = 9

Show how the pages are mapped to three frames using the Least Recently Used (LRU) page replacement algorithm.

Queue:

reference string	1	4	2	3	4	4	1	3	2	1
	{1}	{4,1}	{2,4,1}	{3,2,4}	{4,3,2}	{4,3,2}	{1,4,3}	{3,1,4}	{2,3,1}	{1,2,3}
frame 0	1	1	1	3	3	3	3	3	3	3
frame 1		4	4	4	4	4	4	4	2	2
frame 2			2	2	2	2	1	1	1	1
	X	X	X	X			X		X	

PF = 6