Solution of Exercise Sheet 5

Exercise 1 (Memory Management)

1. Mark memory management methods that cause internal fragmentation to occur.

   ☒ Static partitioning ☐ Dynamic partitioning ☒ Buddy memory allocation

2. Mark memory management methods that cause external fragmentation to occur.

   ☐ Static partitioning ☒ Dynamic partitioning ☒ Buddy memory allocation

3. Explain how external fragmentation can be fixed.

   By defragmentation. For virtual memory, external fragmentation is irrelevant.

4. Mark the memory management method that searches in the entire address space for the block, which fits best to satisfy the request.

   ☐ First Fit ☐ Next Fit ☒ Best fit ☐ Random

5. Mark the memory management concept that searches for the first free block that satisfies the request, starting from the beginning of the address space.

   ☒ First Fit ☐ Next Fit ☐ Best fit ☐ Random

6. Mark the memory management concept that fragments quickly the large area of free space at the end of the address space.

   ☐ First Fit ☒ Next Fit ☐ Best fit ☐ Random

7. Mark the memory management concept that selects randomly a free and appropriate block.

   ☐ First Fit ☐ Next Fit ☐ Best fit ☒ Random

8. Mark the memory management concept that searches for a free block, starting from the latest allocation.

   ☐ First Fit ☒ Next Fit ☐ Best fit ☐ Random

9. Mark the memory management concept that produces many mini-fragments and is slow.

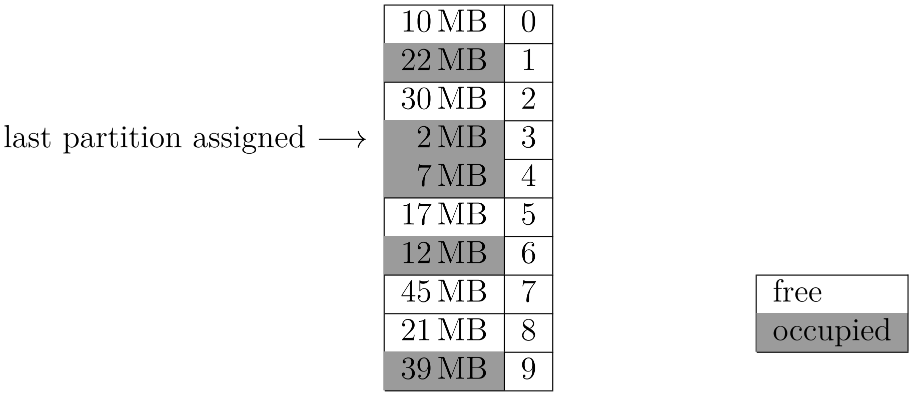
   ☐ First Fit ☐ Next Fit ☒ Best fit ☐ Random

10. Static partitioning can only be implemented using partitions of equal size.

☐ True ☒ False

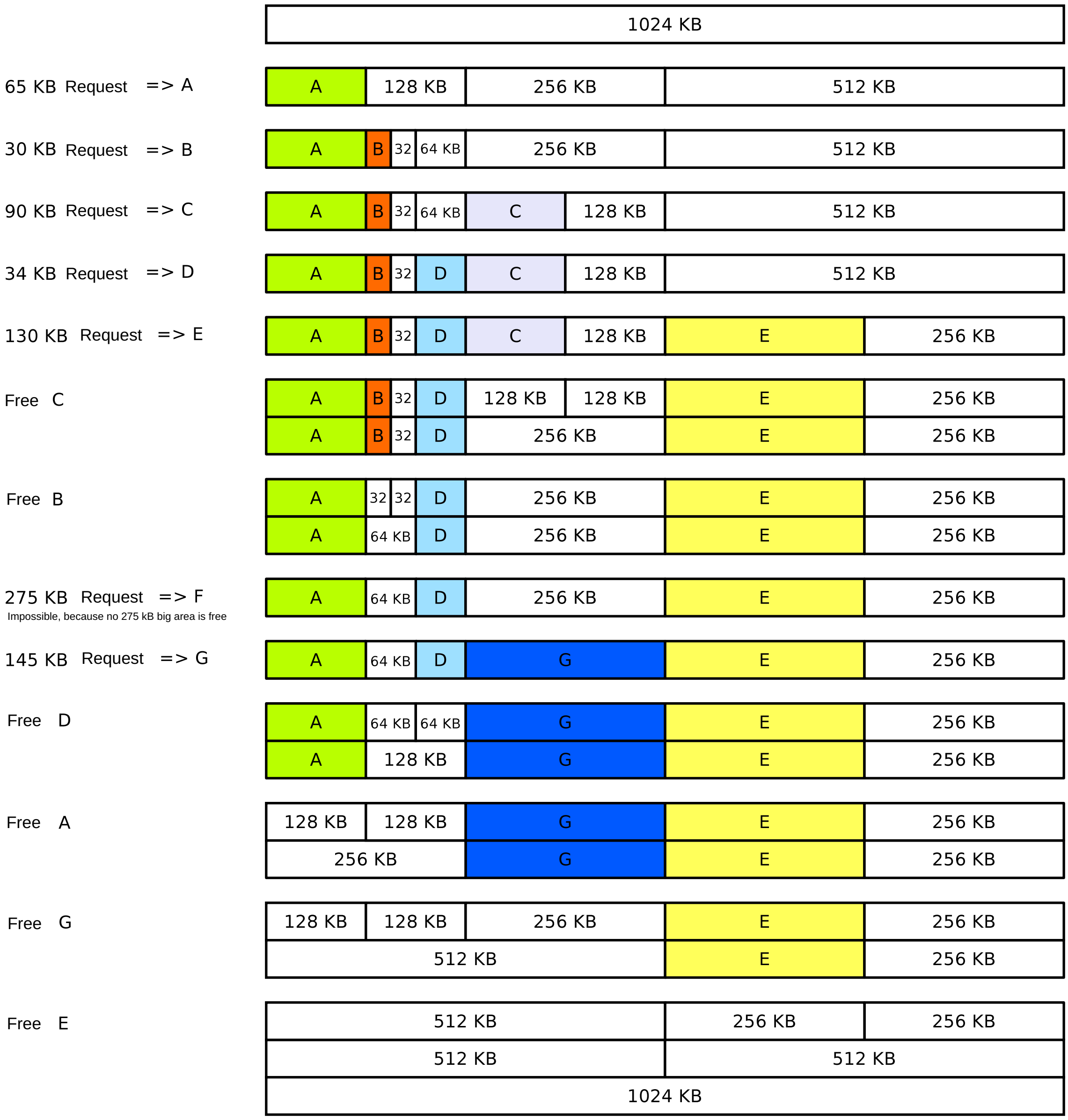
11. The following memory area belongs to a memory with dynamic partitioning. For each of the three algorithms, First Fit, Next Fit, and Best Fit, specify the number of the free partition that the corresponding algorithm uses to insert a process that requires 21 MB of memory.

    a) First Fit: **2** b) Next Fit: **7** c) Best Fit: **8**



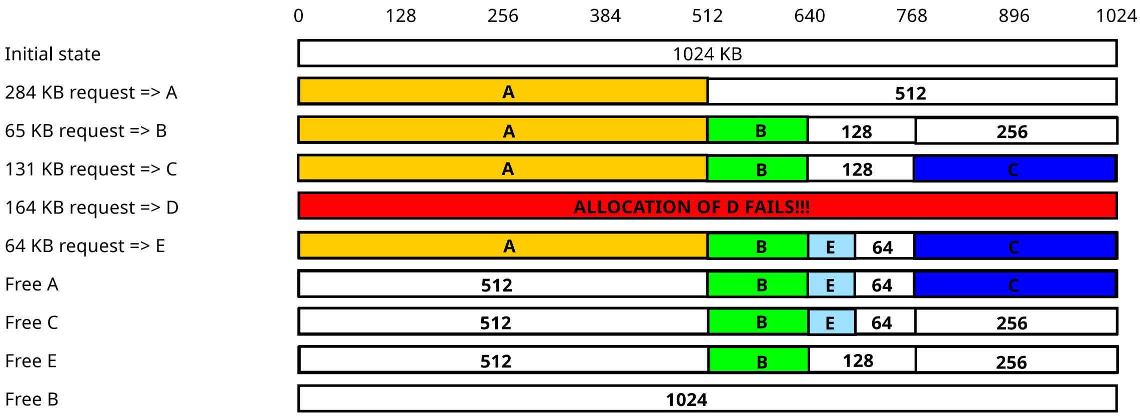
Exercise 2 (Buddy Memory Allocation)

The Buddy method for allocating memory to processes shall be used for a memory with a capacity of 1024 kB. Perform the provided operations and give the occupancy state of the memory after each operation.



Exercise 3 (Buddy Memory Allocation)

Apply the Buddy Allocation algorithm to the memory depicted in the diagram.



Exercise 4 (Real Mode and Protected Mode)

1. Describe the functioning of the real mode.

   Each process can access the entire memory, which can be addressed.

2. Explain why it is impossible to use real mode for multitasking operation mode.

   It provides no memory protection.

3. Describe the functioning of the protected mode.

   Each process can only access its own virtual memory. Virtual memory addresses translates the CPU with the MMU into physical memory addresses.

4. Describe what virtual memory is.

   Each process has a separate address space. This address space is an abstraction of the physical memory. It implements virtual memory. It consists of logical memory addresses, which are numbered from address 0 upwards and it is independent from the storage technology used and the existing expansion options.

5. Explain, why virtual memory helps to better utilize the main memory.

   Processes do not need to be located in one piece inside the main memory. Therefore, the external fragmentation of the main memory is not a problem.

6. Describe what mapping is.

   The virtual memory is mapped to the physical memory.

7. Describe what swapping is.

The process of relocating data from the main memory to the SSD/HDD and back.

8. Name the component of the CPU that is used to implement virtual memory.

   Memory Management Unit (MMU).

9. Describe the function of the component from subtask 8.

   Virtual memory addresses are translated into physical memory addresses by the CPU using the MMU.

10. Describe the virtual memory concept called paging.

    Virtual pages of the processes are mapped onto physical pages in the main memory. All pages have the same length. The page length is usually 4 kb. The operating system maintains a page table for each process. This page table indicates where the individual pages of the process are located. Processes only work with virtual memory addresses. Virtual memory addresses consist of two parts. The higher-value part contains the page number. The lower-value part contains the offset (address within a page). The length of the virtual addresses is architecture-dependent and, therefore, 16, 32, or 64 bits.

11. Describe where internal fragmentation occurs in paging.

    Only in the final page of a process.

12. Give the maximum number of memory addresses a 16-bit computer system can address.

    2^{16} *addresses.*

13. Give the maximum number of memory addresses a 32-bit computer system can address.

    2^{32} *addresses.*

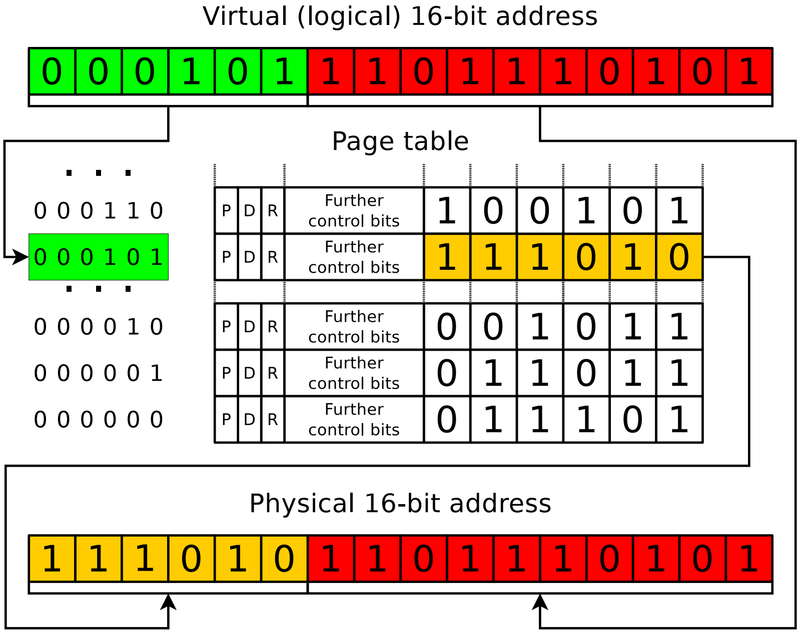
14. Give the maximum number of memory addresses a 64-bit computer system can address.

    It depends on the address bus width. The address bus of modern CPUs (e.g., AMD64, RISC V) has 48 lines. Accordingly, the address space is 2^{48} *addresses..*

15. Explain why multi-level paging is used in 32-bit and 64-bit systems, rather than single-level paging.

    On 32-bit operating systems with a 4 kB page size, the page table of each process can be 4 MB in size. On 64-bit operating systems, the page tables can be considerably larger. Multilevel paging reduces main memory usage since individual pages of the different levels can be relocated into the swap memory to free up memory capacity in the main memory.

16. Calculate the 16-bit physical memory address using single-level paging address conversion. Fill in the individual bits in the 16-bit physical address.



17. Explain the purpose of the Page-Table Base Register (PTBR).

    It stores the address where the page table of the current process starts.

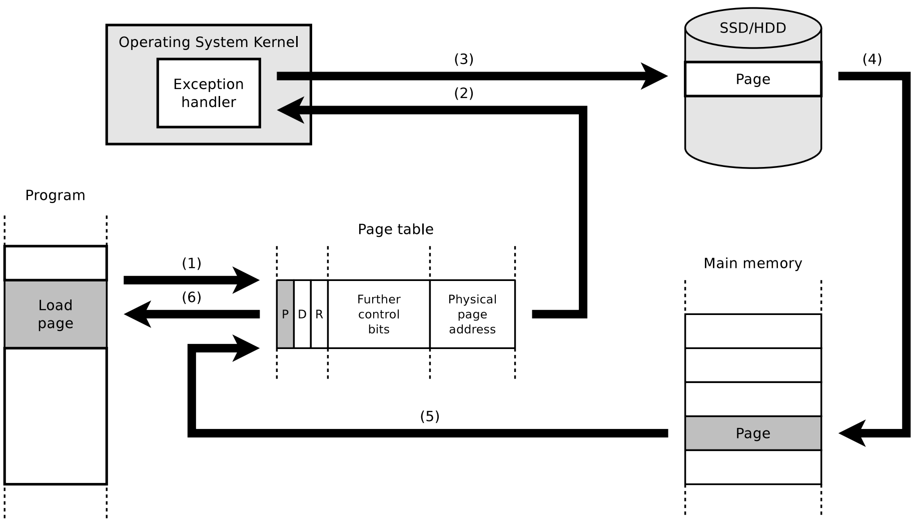
18. Explain the purpose of the Page-Table Length Register (PTLR).

    It stores the length of the page table of the current process.

19. Explain the event that causes a page fault exception.

    A process tries to access a page, which is not located in the physical main memory.

20. The diagram shows a page fault exception. Describe the process stages.



(1) A process tries to request a page, which is not located in the physical main memory

(2) A software interrupt (exception) is triggered to switch from user mode to kernel mode

(3) allocate the page by using the controller and the device driver on the swap memory (SSD/HDD)

(4) copy the page into a free page of the main memory

(5) update the page table

(6) return control to the process

21. Explain what an access violation exception or general protection fault exception causes to occur.

    A process tried to access a virtual memory address, which it is not allowed to access.

22. Describe the consequence (effect) of an access violation exception or general protection fault exception.

    In some legacy Windows operating systems, segmentation faults often caused system crashes and resulted in a blue screen. In Linux, the signal `SIGSEGV` is returned as a result.

23. Describe the content of the kernelspace.

    The kernelspace contains the operating system kernel and kernel extensions (drivers).

24. Describe the content of the userspace.

    The userspace contains the currently running process, which is extended with swap memory (Windows: page file).

Exercise 5 (Memory Management)

Please mark for each one of the following statements, whether the statement is true or false.

1. Real mode is suited for multitasking systems.

   ☐ True ☒ False

2. In protected mode, each process is executed in its own copy of the physical address space, which is protected from other processes.

☒ True ☐ False

3. When static partitioning is used, internal fragmentation occurs.

   ☒ True ☐ False

4. When dynamic partitioning is used, external fragmentation cannot occur.

   ☐ True ☒ False

5. With paging, all pages have the same length.

   ☒ True ☐ False

6. One advantage of long pages is little internal fragmentation.

   ☐ True ☒ False

7. A drawback of short pages is that the page table gets bigger.

   ☒ True ☐ False

8. When paging is used, the MMU translates the logical memory addresses into physical memory addresses.

   ☒ True ☐ False

9. Modern operating systems (for x86) operate in protected mode and use only paging.

   ☒ True ☐ False

Exercise 6 (Page Replacement Strategies)

1. Why is it impossible to implement the optimal replacement strategy OPT?

   Because it is not possible to predict the future and therefore the future request sequence is unknown.

2. Perform the access sequence with the replacement strategies Optimal, LRU, LFU and FIFO once with a cache with a capacity of 4 pages and once with 5 pages. Also calculate the hit rate and the miss rate for all scenarios.

Optimal replacement strategy (OPT):

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	2	1	1	1	1
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3:			5	5	2	2	2	2	2	2	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 4:				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3:			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 4:				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
Page 5:					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1

Hitrate: 17/24 = 0.7083333
Missrate: 7/24 = 0.2916666

Replacement strategy Least Recently Used (LRU):

Requests:	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
Page 1:	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 2:		3	3	3	3	3	3	3	3	3	5	5	5	5	5	5	5	5	5	5	1	1	1	1
Page 3:			5	5	5	5	5	5	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
Page 4:				4	4	4	4	4	4	0	0	0	0	0	0	0	0	0	0	2	2	2	2	5
Queue:	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
		1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4
			1	3	5	5	2	4	3	2	1	0	0	3	5	0	4	3	5	4	3	2	1	3
				1	3	3	5	5	4	3	2	1	1	1	3	5	0	0	0	5	4	4	2	1

Hitrate: 11/24 = 0.4583333%
Missrate: 13/24 = 0.5416666%

Requests:	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
Page 1:	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
Page 3:			5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
Page 4:				4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Page 5:					2	2	2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	4	4	4
Queue:	1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4	5
		1	3	5	4	2	4	3	2	1	0	5	3	5	0	4	3	5	4	3	2	1	3	4
			1	3	5	5	2	4	3	2	1	0	0	3	5	0	4	3	5	4	3	2	1	3
				1	3	3	5	5	4	3	2	1	1	1	3	5	0	0	0	5	4	4	2	1
					1	1	1	1	5	4	3	2	2	2	1	1	1	1	1	0	5	5	5	2

Hitrate: 14/24 = 0.583333%
Missrate: 10/24 = 0.416666%

Replacement strategy Least Frequently Used (LFU):

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1_1	1_1	1_1	1_1	2_1	2_1	2_1	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_3	2_3	2_3	2_3	2_3
Page 2:		3_1	3_1	3_1	3_1	3_1	3_2	3_2	3_2	3_2	3_2	3_3	3_3	3_3	3_3	3_4	3_4	3_4	3_5	3_5	3_5	3_6	3_6	3_6
Page 3:			5_1	5_1	5_1	5_1	5_1	5_1	1_1	0_1	5_1	5_1	5_2	5_2	5_2	5_2	5_3	5_3	5_3	5_3	5_3	5_3	5_3	5_4
Page 4:				4_1	4_1	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_2	0_1	4_1	4_1	4_1	4_2	4_2	4_2	1_1	1_1	4_1	4_1

Hitrate: 12/24 = 0.5
Missrate: 12/24 = 0.5

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1_1	1_1	1_1	1_1	1_1	1_1	1_1	1_1	1_2	1_2	1_2	1_2	1_2	0_1	0_1	0_1	0_1	0_1	0_1	0_1	1_1	1_1	1_1	1_1
Page 2:		3_1	3_1	3_1	3_1	3_1	3_2	3_2	3_2	3_2	3_2	3_3	3_3	3_3	3_3	3_3	3_4	3_4	3_5	3_5	3_5	3_6	3_6	3_6
Page 3:			5_1	5_1	5_1	5_1	5_1	5_1	5_1	0_1	5_1	5_1	5_2	5_2	5_2	5_2	5_3	5_3	5_3	5_3	5_3	5_3	5_3	5_4
Page 4:				4_1	4_1	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_2	4_3	4_3	4_3	4_4	4_4	4_4	4_4	4_4	4_5	4_5
Page 5:					2_1	2_1	2_1	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_2	2_3	2_3	2_3	2_3	2_3

Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

Replacement strategy FIFO:

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	5
Page 2:		3	3	3	3	3	3	3	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
Page 3:			5	5	5	5	5	5	5	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2
Page 4:				4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	1	1	1	1

Hitrate: 11/24 = 0.4583333
Missrate: 13/24 = 0.5416666

Requests:	**1**	**3**	**5**	**4**	**2**	**4**	**3**	**2**	**1**	**0**	**5**	**3**	**5**	**0**	**4**	**3**	**5**	**4**	**3**	**2**	**1**	**3**	**4**	**5**
Page 1:	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Page 2:		3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	1	1	1	1
Page 3:			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	3	3	3
Page 4:				4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5
Page 5:					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Hitrate: 15/24 = 0.625
Missrate: 9/24 = 0.375

3. Describe the key message of Laszlo Belady's anomaly.

   FIFO produces worse results for certain access patterns with increased memory.

4. Show Belady's anomaly by performing the access sequence with the replacement strategy FIFO once with a cache with a capacity of 3 pages and once with 4 pages. Also calculate the hit rate and the miss rate for both scenarios.

Requests:	**3**	**2**	**1**	**0**	**3**	**2**	**4**	**3**	**2**	**1**	**0**	**4**
Page 1:	**3**	3	3	**0**	0	0	**4**	4	4	4	4	4
Page 2:		**2**	2	2	**3**	3	3	3	3	**1**	1	1
Page 3:			**1**	1	1	**2**	2	2	2	2	**0**	0

Hitrate: 3/12 = 25%
Missrate: 9/12 = 75%

Requests:	**3**	**2**	**1**	**0**	**3**	**2**	**4**	**3**	**2**	**1**	**0**	**4**
Page 1:	**3**	3	3	3	3	3	**4**	4	4	4	**0**	0
Page 2:		**2**	2	2	2	2	2	**3**	3	3	3	**4**
Page 3:			**1**	1	1	1	1	1	**2**	2	2	2
Page 4:				**0**	0	0	0	0	0	**1**	1	1

Hitrate: 2/12 = 16.66%
Missrate: 10/12 = 83.33%

Exercise 7 (Time-based Command Execution, Sorting, Environment Variables)

1. Create in your home directory a directory `NotImportant` and write a cron job, which erases the content of the directory `NotImportant` every Tuesday at 1:25 clock am.

   The output of the command should be appended to a file `EraseLog.txt` in your home directory.

   ```
   $ mkdir ~/NotImportant
   $ crontab -e
   ```

   Insert these lines:

   ```
   25 1 * * 2 rm -rfv /home/USERNAME/NotImportant/* >>
   /home/USERNAME/EraseLog.txt
   ```

2. Write a cron job, which appends a line at a file `Datum.txt` with the following format (but with the current values) every 3 minutes between 14:00 to 15:00 clock on every Tuesday in the month of November:

```
Heute ist der 30.10.2008
Die Uhrzeit ist 09:24:42 Uhr
****************************
```

```
$ crontab -e
```

Insert these lines:

```
*/3 14 * 11 * date +"Heute ist der \%x\%nDie Uhrzeit ist
\%H:\%M:\%S Uhr\%n****************************" >> Datum.txt
```

3. Write an at-job, which outputs at 17:23 today a list of the running processes.

> *You may have to install the command line tool* `at` *first.*
> *With Debian/Ubuntu this works with:*
> *`$ sudo apt update && sudo apt install at`*
> *With CentOS/Fedora/RedHat this works with:*
> *`$ sudo yum install at`*

```
$ at 1723 today
```

Insert these lines:

```
ps -r
```

4. Write an at-job, which outputs at December 24th at 8:15 am the text „It's christmas!“

```
$ at 0815 DEZ 24
```

Insert these lines:

```
echo "It's christmas!"
```

5. Create in your home directory a file `Kanzler.txt` with the following content:

```
Willy     Brandt    1969
Angela    Merkel    2005
Gerhard   Schröder  1998
KurtGeorg Kiesinger 1966
Helmut    Kohl      1982
Konrad    Adenauer  1949
Helmut    Schmidt   1974
Ludwig    Erhard    1963
```

```
$ echo "Willy     Brandt    1969" >> ~/Kanzler.txt
$ echo "Angela    Merkel    2005" >> ~/Kanzler.txt
```

```
$ echo "Gerhard   Schröder  1998" >> ~/Kanzler.txt
$ echo "KurtGeorg Kiesinger 1966" >> ~/Kanzler.txt
$ echo "Helmut    Kohl      1982" >> ~/Kanzler.txt
$ echo "Konrad    Adenauer  1949" >> ~/Kanzler.txt
$ echo "Helmut    Schmidt   1974" >> ~/Kanzler.txt
$ echo "Ludwig    Erhard    1963" >> ~/Kanzler.txt
```

6. Print out the file `Kanzler.txt` sorted by the first names.

```
$ sort ~/Kanzler.txt
```

7. Print out the file `Kanzler.txt` sorted by the third letter of the last names.

```
$ sort -k+2.4 ~/Kanzler.txt
```

8. Print out the file `Kanzler.txt` sorted by the year of the inauguration.

```
$ sort -k3 ~/Kanzler.txt
```

9. Print out the file `Kanzler.txt` backward reverse sorted by the year of the inauguration and redirect the output into a file `Kanzlerdaten.txt`.

```
$ sort -k3 -nr ~/Kanzler.txt > ~/Kanzlerdaten.txt
```

10. Create with the command `export` an environment variable `VAR1` and assign it the value `Testvariable`.

```
$ export VAR01=Testvariable
```

11. Print out the value of `VAR1` in the shell.

```
$ printenv VAR01
```

12. Erase the environment variable `VAR1`.

```
$ unset VAR01
```