ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communications
CS–202 Computer Systems
Argyraki K., Kashyap S. & Chappelier J.-C.

NOM : Hanon Ymous
(000000) #0000
Seat #: 0

CS–202 COMPUTER SYSTEMS

Midterm

April 7th, 2025

INSTRUCTIONS (please read carefully)

IMPORTANT! Please strictly follow these instructions, otherwise your exam may be canceled.

1. You have one hour and forty-five minutes to complete this examination (1:15–3:00pm).
2. You must **use black or dark blue ink**, neither pencil nor any other color.
3. This is closed book exam.
   Personal notes, two times dual-sided A4 sheets (4 sides in total), allowed.

   On the other hand, you may not use any personal computer, mobile phone or any other electronic equipment.
4. Answer the questions directly on the exam sheet; do not attach any additional sheets; only this document will be graded.
5. Carefully and *completely* read each question so as to do only what we actually ask for. If the statement seems unclear, or if you are in any doubt, ask one of the assistants for clarification.
6. The exam consists of four independent exercises, which can be addressed in any order, but which do not score the same (points are indicated, the total is 105 points); all exercises count for the final grade.

LEAVE THIS EMPTY				
Question 1	Question 2	Question 3	Question 4	**TOTAL**
20	25	25	35	105

Question 1 Accessing a file [20 points]

Consider a file system with the following properties:

- block size of 4KiB (4096 bytes);
- each inode contains metadata and 10 direct pointers to blocks that contain data.

Consider the following C program excerpt:

```
1   ...
2   const char* filename = "/foo/bar/testfile.txt";
3
4   int main()
5   {
6       const size_t N = ...;
7       char buffer[N];
8
9       for (size_t i = 0; i < N; ++i) { buffer[i] = ...; }
10
11      int fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0644);
12      ...
13
14      ssize_t bytes_written = write(fd, buffer, N);
15      if (bytes_written < 0) {
16          return 1;               // Write failed.
17      }
18      ...
19
20      close(fd);
21      return 0;
22  }
```

Suppose you compile it and invoke it on your computer. Assume you have all the necessary permissions and that file "`/foo/bar/testfile.txt`" already exists.

① **[1 point]** How many processes and how many threads does the Operating System (OS) of your computer create as a result of you invoking this program?

② **[2 points]** At the moment when the CPU is executing the first instruction of this program:
(a) Where is the Instruction Pointer (IP) register pointing at?
(b) In what mode/level is the CPU?

③ **[2 points]** Identify all the calls that cause the CPU to switch from one mode/level to a different one while executing this program. Next to each call, say why it causes a switch in one sentence.

④ **[10 points]** Let's define a "disk access" as an instance where the CPU tells the disk controller that it wants to read or write a sequence of bytes from/into a contiguous area of the disk.

Assuming the `if` statement on line `15` is false, how many disk accesses does this program trigger? For each disk access, state what is read from or writen to the disk.

⑤ **[5 points]** Given the properties of this file system, what is the smallest value of `N` that will necessarily make the `if` statement on line `15` true? **Justify** your answer in 1–3 sentences.

Answers:

more space to answer at the back ☞

Answers to Question 1 continued:

Answers to Question 1 continued:

continues on back ☞

Question 2 Memory virtualization [25 points]

Consider a computer system that uses paging for memory virtualization and has the following properties:

- each page consists of 128 bytes;
- each virtual address consists of 10 bits in total
  (including the bits used to identify the offset within each page).

① **[1 point]** How many bits are used to identify the offset within each page?
Justify your answer in 1–2 sentences.

② **[1 point]** How many bits are used to index into a page table?
Justify your answer in 1–2 sentences.

③ **[1 point]** Assume linear (single-level) page tables.
Compute the minimum size of each page table.

④ **[2 points]** With the information you have until this point, can you compute the size of the computer's physical memory? If yes, compute it. If not, explain why not in 1–2 sentences.

⑤ **[10 points]** The following table shows you the inputs and outputs of the Memory Management Unit (MMU) at different points in time (input and output are in binary):

Time tick	MMU Input	MMU Output
0	0010000000	00110000000
1	0110000111	00010000111
2	0001010100	01011010100
3	1000011001	00100011001
4	0100111000	01100111000
5	1100011011	01110011011
6	1110000000	01000000000
7	1011000100	00001000100
8	0001001100	10101001100
9	0011110000	10001110000
10	0011010000	10001010000

From time tick 0 up to and including time tick 7, the CPU runs the same process, P1.
Draw as much as you can of P1's page table.

⑥ **[5 points]** If all processes running in this computer system have **similar memory images** as process P1, is there any benefit in using two-level page tables (as opposed to linear page tables)?
Justify your answer in 1-3 sentences.

⑦ **[5 points]** By looking at the table above, can you tell whether the CPU continues to run process P1 or switches to a different process from time tick 8 and onward?

Suppose the timer interrupt occurs every X time ticks. Do you have enough information to compute an upper bound for X?
If yes, compute it. If not, explain what extra assumptions you would need to make about process P1 in order to compute it.

Do NOT write anything here!

Answers:

more space to answer at the back ☞

Answers to Question 2 continued:

Answers to Question 2 continued:

continues on back ☞

Question 3 Forking and scheduling [25 points]

Consider an OS scheduler that follows a Multi-Level Feedback Queue (MLFQ) policy with the following properties:

- there are two priority levels, H (higher) and L (lower);
- both levels have a time slice of 2 time ticks;
- when a thread arrives, it is placed in level H;
- if a thread is in level H and exhausts its time slice, it is demoted to level L;
- all threads are boosted to level H every 5 time ticks (so, at time tick 5, 10, 15, etc.).

Consider the C program excerpt provided below. Next to each statement, you see how long it takes to execute the corresponding instructions. If there is no timing information written next to a statement, assume that the corresponding instructions take 0 time.

```
1   ...
2   int main()
3   {
4      int pid1 = fork();   // 1 time tick.
5      if (pid1 == 0) {
6         ...               // 1 time tick. Initiates I/O that takes 1000 time ticks.
7         return 0;
8      }
9      int pid2 = fork();   // 1 time tick.
10     if (pid2 == 0) {
11        ...               // 1 time tick. Initiates I/O that takes 1000 time ticks.
12        return 0;
13     }
14     waitpid(pid1, ...);  // 1 time tick.
15     waitpid(pid2, ...);  // 1 time tick.
16     return 0;
17  }
```

Suppose you compile this program and invoke it on your computer.
The CPU starts executing the first instruction at time tick 0.
The two I/O operations do not block one another (they can proceed at the same time).

① **[2 points]** How many processes and how many threads does your OS create as a result of you invoking this program? Draw the parent/child process graph.

② **[15 points]** Show which thread(s) are in level H and L, which thread(s) are blocked, which thread(s) are ready, and which thread is running at each time tick during the first 6 time ticks. Any category (including the "running" category) may be empty.
Use Table 1 on the next page.

③ **[2 points]** Compute the average turnaround time from time tick 0 until when all threads have terminated.

④ **[6 points]** Would this metric change significantly (and if yes, by how much) if the OS scheduler followed a Shortest Time to Completion First (STCF) policy?
Justify your answer in 1–3 sentences.
Hint: you do not need to draw a table like Table 1 in order to answer this subquestion.

Answers:

Time tick	0	1	2	3	4	5
Arriving threads						
Threads in H						
Threads in L						
Blocked threads						
Ready threads						
Running thread						

Table 1: Your answer to sub-question ②.

continues on back ☞

Question 4 A bit of C [35 points]

This is a set of three independent questions about C programming.
For your answers, you can use whatever C standard up to (and including) C23.

4.1 A bit of memory [13.5 points]

Consider the following C program (which compiles):

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void f(const char* tab[], size_t n)
{
  char* new = calloc(3,1);
  char* it = new;
  *new++ = **tab;
  *new = tab[n-1][4];
  printf("\"%s\"\n", it);
  free(it);
}

int main(void)
{
  const char* hello = "Hello CS-202!";
  size_t p = 4;
  char c = 'x';
  const char* some[] = { hello + p, strstr(hello, "-2"), &c };

  f(some, 2);

  return 0;
}
```

As a reminder, the function

```
char* strstr(const char* haystack, const char* needle);
```

returns (a pointer to) the position of the substring `needle` inside the string `haystack`.

① **[1 point]** For each of the 8 variables/parameters (`tab`, `n`, `new`, `it`, `hello`, `p`, `c` and `some`), put their name in the appropriate column corresponding to where *they* stay in memory:
(maybe **read the next question first**)

stack	heap	text or data segment

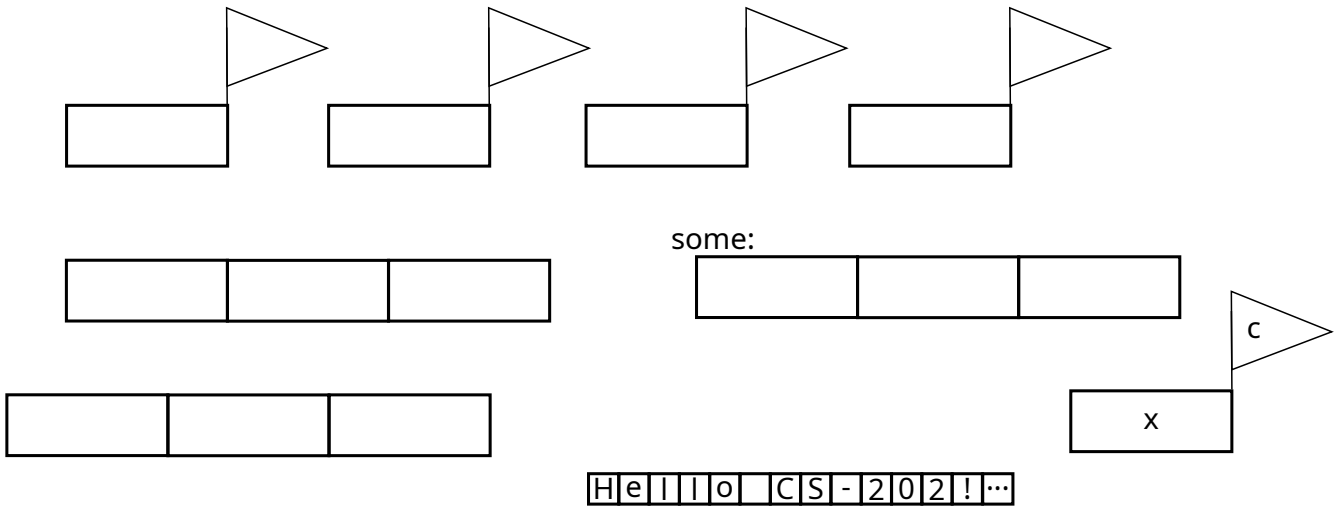
② **[3 points]** For each of the following pointers, say whether they **point to** stack, heap or text/data segment:

`some[0]`:	`hello`:
`some[1]`:	`new`:
`some[2]`:	`it`:

③ **[4 points]** What does the program print? If there are some unknown values, simply use a question mark (`?`) for each of them.
Briefly explain your answer.

Answer:

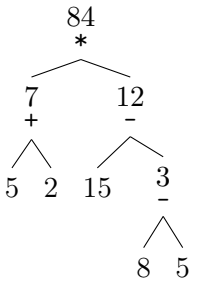
④ **[5.5 points]** Using the template below, draw an abstract picture of the memory state for the six variables (and their relations) `it`, `new`, `tab`, `some`, `c` and `hello` when the `printf()` is done.
There may be more cells than needed; and if you miss cells, simply add your own.



continues on back ☞

4.2 A bit of arithmetic [14.5 points]

Arithmetic computations can be represented in the form of a tree, each node of which contains an operator, a value, and a left and a right computation (= subtrees).
For instance, the computation $(5 + 2) \times (15 - (8 - 5))$ could be represented as the tree on the right, where for inner nodes, the corresponding value has been displayed above the operator.
For pure values, i.e. leaves of the tree, the "operator" is simply a special operator named "value" and their left and right computations are simply empty.



4.2.1 Types [2.5 points]

Here first define an enumerated type named `Operation` with the values: `ADD`, `SUB`, `MULT`, `DIV` and `VALUE`.

Then define a type `Tree` which contains: an `Operation`, a value (`unsigned int`) and two "`Tree`s", `left` and `right`. It's up to you to choose the appropriate type for `left` and `right` (the reason why we quoted it).

Answer:

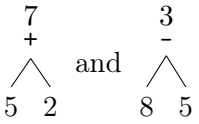
4.2.2 Values [4 points]

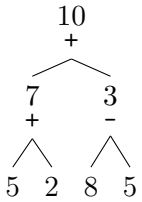
Define a function `value()` that takes a `unsigned int` and returns a pointer to a newly allocated `Tree` which contains only a value, i.e. it's a leaf: operation is `VALUE`, the value is the parameter of the function and `left` and `right` are "empty trees" (it's up to you to choose how to represent that according to your former choices).
This function returns `NULL` in case of error.

Answer:

4.2.3 Addition [5 points]

Define a function `add()` that takes two pointers to `Tree` and returns a pointer to a newly allocated `Tree` which represents the addition of the two `Tree` received as parameters.
It returns `NULL` in case of error.

For instance, if this function receives the trees 

and it will then return the tree 

Answer:

4.2.4 Garbage collector [3 points]

Define a function `free_tree()` that take a pointer to `Tree` and releases *all* the memory that has been allocated for that tree (assuming it was itself dynamically allocated).
This function does not return anything.

Answer:

continues on back ☞

4.3 A bit of strings [7 points]

Using pointer arithmetic, write a *function*

```
char* normalize(const char* string);
```

that takes a(n immutable) string and returns a copy of it, where all whitespaces are replaced with `'_'` (underscore) and all characters are put in lowercase.
This function returns `NULL` in case of error.

To test if a character is a whitespace, you must use the standard function

```
int isspace(int c);
```

and to get the lowercase of a character, you must use the standard function

```
int tolower(int c);
```

This function is robust (returns the same character if it has no lowercase equivalent).

Note: in C, a `char` can silently be cast to and cast from an `int`.

Answer: