# COMP 3511 Operating System (Fall 2022)

## Midterm Exam

Date: 01-November-2022 (Tuesday)

**Time: 19:00 – 21:00 (2 hours)**

| | |
|---|---|
| Name<br><br>(Write the name printed on your Student ID card) | **Solution (released before the midterm exam paper checking)**<br><br>**We may further update it after the midterm paper checking** |
| Student ID | |
| ITSC email | @connect.ust.hk |

Exam format and rules:

- It is an open-book, open-notes exam (Reference: Chapter 1).
- No electronic devices, except a calculator with basic functions, are allowed.
- Other details are sent to students via exam related emails

| Question | Marks |
|---|---|
| Q1 – Multiple Choices | /25 |
| Q2 - Process and Thread | /25 |
| Q3 - CPU Scheduling | /36 |
| Q4 – Synchronization | /14 |
| **Total** | **/100** |

## Problem 1 Multiple Choices [25 Points]

Write down your answers in the boxes below:

| MC1 | MC2 | MC3 | MC4 | MC5 | MC6 | MC7 | MC8 | MC9 | MC10 |
|---|---|---|---|---|---|---|---|---|---|
| **B** | **A** | **D** | **C** | **A** | **D** | **B** | **C** | **D** | **A** |

| MC11 | MC12 | MC13 | MC14 | MC15 | MC16 | MC17 | MC18 | MC19 | MC20 |
|---|---|---|---|---|---|---|---|---|---|
| **C** | **D** | **B** | **C** | **D** | **A** | **B** | **D** | **D** | **A** |

| MC21 | MC22 | MC23 | MC24 | MC25 |
|---|---|---|---|---|
| **C** | **C** | **B** | **B** | **C** |

**[Introduction]**
MC1. Which of the following statements is NOT true on memory hierarchy?
A. Registers run faster than cache
B. The hard-disk drives (HDDs) run faster than nonvolatile memory (NVM) devices
C. Memory is more expensive per byte than any secondary storage devices
D. The capacity of cache is usually much smaller than that of memory
Answer: B

MC2. Which of the following statements on interrupt is NOT true?
A. I/O devices using DMA technique do not generate interrupt
B. Interrupt can be generated by software
C. CPU checks whether there is an interrupt between the executions of instructions
D. All modern computer systems are interrupt-driven
Answer: A

MC3. Which component below is part of the *von Neumann* architecture?
A. Arithmetic Logic Unit or ALU
B. Registers including program counter (PC) and instruction register (IR)
C. Memory along with cache
D. All of the above
Answer: D

**[Operating System Structures]**
MC4. Which of the following statements is NOT true about APIs?
A. It allows running programs to request services from operating systems
B. It hides the complex details in system calls.
C. It intercepts function calls and invokes the necessary system calls

D. It provides program portability
Answer: C (Slide )
The system call interface intercepts function calls in APIs and invokes the necessary system calls within an operating system

MC5. Which of the following components is NOT part of Darwin?
A. Core services for cloud computing and database
B. Mach kernel and traps
C. IPC mechanisms
D. Memory management modules
Answer: A

MC6. Which of the following is TRUE in a loadable kernel module or LKM design?
A. LKM combines the benefits of the layered and microkernel design techniques
B. There is no need to either recompile or reboot the kernel
C. LKM enables functionalities be added to and removed from the kernel while it is running
D. All of the above
Answer: D

MC7. Which of the following statements is NOT true in a microkernel design?
A. New services can be added to user space without modification on the kernel
B. The kernel is smaller, so the performance of microkernels is better
C. It is more secure and reliable
D. It is easier to port or migrate to a new architecture
Answer: B

**[Processes]**
MC8. Which of the following statements is NOT true on dual-model operation?
A. It distinguishes when the system is running user codes or kernel codes
B. It allows OS to protect itself and other system components
C. It allows some instructions can only be executed in the kernel mode while some other instructions can only be executed in the user mode
D. It ensures that certain instructions can never be executed in the user mode
Answer: C (Slides 3.21-3.22)

MC9. Which of the following statements is TRUE on the possible resource sharing between a parent process and a child process?
A. A parent process and a child process may share all resources
B. A parent process and a child process may share a subset of resources
C. A parent process and a child process may share no resources
D. All of the above
Answer: D (Slide 3.27)

MC10. Which of the following statements is NOT true about a zombie process?
A. A parent process waiting for a terminated child is called a zombie
B. All processes transition into this zombie state before termination
C. A zombie process must be already terminated

D. Most of the resources allocated to a zombie process such as memory, open files, and I/O buffers have been reclaimed by operating system
Answer: A (Slide 3.39)
A process that has terminated, but whose parent has not yet called wait(), is known a zombie process – the terminated process, not the parent, is referred as a zombie

MC11. What is the total number of processes in the following C code if all fork() are successful?

```
int main() {
  if (fork()>0) {
    if (fork() == 0)
        fork();
  }
  return 0;
}
```

A. 2
B. 3
C. 4
D. 5
Answer: C
There are two processes at the end of the first fork, but only one of them proceeds to execute the second fork (now total 3processes), and one of which executes the third fork, so total 4 processes.

MC12. What of the following statements is NOT true in a client-server communication using socket?
A. A server's machine address (i.e., IP address) and port number must be known
B. The 4-tuple (source IP address and port number, destination IP address and port number) uniquely determines a pair of communications
C. Only a client can initiate a communication
D. None of the above
Answer: D

**[Threads]**
MC13. Within a multithreaded process, which of the following components is NOT shared by threads within a process?
A. The code segment
B. The stack
C. The global variable
D. The opened files
Answer: B (Slide 4.6, 4.15-4.16)

MC14. Which of the following statements on user thread is TRUE?
A. User threads can be scheduled to run on CPU
B. The operating system manages and schedules user threads
C. User threads are created and managed by threading libraries
D. User threads enable multicores or multiprocessors to run a program in parallel
Answer: C (4.20)

Threads libraries provide APIs for creating and managing user threads in programs

MC15. According to Amdahl's Law, what is a possible combination of percentage of parallel components and the number of processing cores if the speedup gain is at most 1.5?
A. percentage of parallel components: 75%, processing cores: 2
B. percentage of parallel components: 75%, processing cores: 3
C. percentage of parallel components: 50%, processing cores: 2
D. percentage of parallel components: 50%, processing cores: 3
Answer: D
speedup of A: 1.6, B: 2, C: 1.33, D: 1.5

MC16. Which of the following statements on clone() is FALSE?
A. clone() duplicates the data structure from the parent like fork()
B. clone() enables a child to share part of the execution context with the parent
C. clone() allows a child to select the extent of sharing with the parent
D. clone() requires loading a new function to execute in a child process.
Answer: A (Slide 4.36)

**[CPU Scheduling]**
MC17. Which of the following statements is NOT true on FCFS vs. RR scheduling?
A. RR is intrinsically more fair than FCFS
B. RR always has a shorter average turn-around time than FCFS
C. FCFS suffers from convoy effect, while RR does not
D. RR always results no worse response time than FCFS
Answer: B

MC18. Which of the following statement is NOT true on SJF vs. SRTF scheduling?
A. SRTF is a preemptive version of SJF
B. SRTF performs no worse than SJF in term of the average waiting time
C. SRTF may have a greater number of context switches than SJF
D. None of the above
Answer: D (5.7)

MC19. Which of the following statements about MLFQ is TRUE?
A. MLFQ produces better response time than RR when the top queue uses the same quantum as that in RR
B. MLFQ delivers similar performance as that of SJF or SRTF
C. MLFQ does not require any prior knowledge in the next CPU burst time
D. All of the above
Answer: D

MC20. Which of the following threads is handled by CPU scheduling?
A. A kernel thread or software thread scheduled to a hardware thread
B. A user thread scheduled to a kernel thread
C. A hardware thread scheduled to a CPU core
D. None of the above
Answer: A (Slide 5.43)

MC21. Which of the following statements is NOT true for earliest-deadline-first (EDF) scheduling algorithm?
A. EDF requires the deadline to be specified upon a process arrival
B. EDF does not require a process to be periodic
C. EDF requires that process time is a constant
D. None of the above
Answer: C (Slide 5.51)

MC22. Suppose that there are five processes arriving at a ready queue at the same time. Their CPU burst times are all 5 time units. Please derive the average waiting times under FCFS and RR with a quantum 2 time units, respectively.
A. FCFS 10 and RR 20
B. FCFS 15 and RR 18
C. FCFS 10 and RR 18
D. FCFS 15 and RR 20
Answer: C
FCFS (0+5+10+15+20)/5 = 10
RR(2) (16+17+18+19+20)/5 = 18

**[Synchronization]**
MC23. Which of the following statements on *Spinlock* is FALSE?
A. Spinlocks incur busy waits
B. Spinlocks are useful in both single-processor and multiprocessor systems
C. Spinlocks is effective when the lock is held by a running process on another CPU
D. Spinlocks waste CPU cycles
Answer: B (Slide 6.22 and 7.11)

MC24. When using semaphores, a process invokes wait() before accessing its critical section, followed by the signal() upon completion of its critical section. What would be a possible outcome if the process first calls signal() before entering its critical section, then calls wait() afterwards?
A. Starvation is possible.
B. Multiple processes can enter the critical sections at the same time
C. Mutual exclusion is still ensured
D. Deadlock is possible
Answer: B

MC25. Suppose the following code ensures that three threads executing in critical section in order, i.e., T1, T2, T3, T1, T2, T3, T1 … What shall be the initial values of the three semaphores S1, S2, and S3, respectively?

```
Thread T1:
while (TRUE) {
   wait (S1);
   Critical Section
   signal (S2);
}
```

```
Thread T2:
while (TRUE) {
   wait (S2);
   Critical Section
   signal (S3);
}
```

```
Thread T3:
while (TRUE) {
   wait (S3);
   Critical Section
   signal (S1);
}
```

A. S1 = 1; S2 = 1; S3 = 1
B. S1 = 0; S2 = 0; S3 = 1
C. S1 = 1; S2 = 0; S3 = 0
D. S1 = 0; S2 = 0; S3 = 0
Answer: C

## Problem 2 Process and Thread [25 Points]

Suppose all `fork()` are successful and necessary header files are included.
Function `fflush(stdout)` is used to print the data in the buffer immediately, so that `fork()` won't copy these unprinted result to the child process.

1) (5 points) Recall from project assignment 1, you are asked to implement a simplified shell program. In this question, you need to fill in four missing lines to complete an example of input and output redirection.

```
int main() {
    // 0: stdin, 1: stdout
    int fin = open("in.txt", O_RDONLY, S_IRUSR | S_IWUSR) ;
    int fout = open("out.txt", O_CREAT | O_WRONLY, S_IRUSR |
S_IWUSR );


    _____________________________________________


    _____________________________________________


    _____________________________________________


    _____________________________________________

    int a, b;
    scanf("%d %d", &a, &b);
    printf("%d", a+b);
    return 0;
}
```

To limit the number of possible answers, you can only use the following 2 system calls: `close` and `dup`. You cannot use other system calls (e.g., `dup2` cannot be used). Here is the expected result after running the program:

| in.txt (this file exists in the same directory as the executable file) | out.txt (this file does not exist before the execution of the program) |
|---|---|
| 2 3 | 5 |

Answer:

```
// close(0); dup(fin); can be swapped with close(1); dup(fout);
// But close must be followed by dup. We cannot have 2 close and then 2 dup
// Other system calls won't be accepted.
```

(1 mark for each line), all correct (+1 additional mark). Missing ; overall -0.5 marks. Some partial credits may be given for incorrect answers.

Possible solution 1:

```
close(0);
dup(fin);
close(1);
dup(fout);
```

Possible solution 2:

```
close(1);
dup(fout);
close(0);
dup(fin);
```

2) (10 points) Consider the following code snippet:

```
#include <unistd.h>
#include <stdio.h>
#include <pthread.h>
#include <sys/wait.h>
int x = 7;
void *substract(void *arg) {
        int *data = arg;
        x -= *data;
        return NULL;
}
int main(int argc, char *argv[]) {
        pid_t pid;
        pthread_t tid;
        pthread_attr_t attr;
        int num = 8;
        printf("ORIGINAL: x = %d \n", x); /* LINE A */
        fflush(stdout); /* clear output buffer */
        pid = fork();
        if (pid > 0) {/* parent process */
                pthread_attr_init(&attr);
                pthread_create(&tid, &attr, substract, &num);
                pthread_join(tid, NULL);
```

```
                printf("THREAD: x = %d \n", x); /* LINE B */
                fflush(stdout);
        }
        else if ( pid == 0 ) {/* child process */
                num = -18;
                substract(&num);
                printf("CHILD: x = %d \n", x); /* LINE C */
                fflush(stdout);
        }
        printf("FINAL: x = %d \n", x); /* LINE D */
        fflush(stdout);
        wait(NULL);
}
```

What is the output of the above program? For each line of output, please justify your answer. You only need to write down one possible answer for this question.

For example, if your program has N lines of output, you should write your answers in the following format:

*Output of line 1 (Explanation for line 1)*
*Output of line 2 (Explanation for line 2)*
…
*Output of line N (Explanation for line N)*

**Answer**: (2 points each)
Note: The output ordering may be different. Pay attention when grading this question
ORIGINAL: x = 7 (original value)
THREAD: x = -1 (thread substracts 8 from the value)
FINAL: x = -1 (thread updates the global value in the parent process x)
CHILD: x = 25 (child process substracts -18 to the value)
FINAL: x = 25 (the remaining code of child process makes no modification to x)

3) (10 points) Consider the following program.

```
int main() {
        int i, n;
        printf("Enter a positive integer n: ");
        scanf("%d", &n);
        for (i = 0; i < n; i++) {
                if (fork() == fork()) {
                        printf("fork() == fork()!\n");
                        fflush(stdout);
                }
        }
        return 0;
}
```

(a) (3 points) How many processes are executed and how many lines of `fork() == fork()` are printed when `n=1, 2, 3`. Write your answer in the following table:

| n | Number of processes | Number of lines |
|---|---|---|
| 1 | 4 | 1 |
| 2 | 16 | 5 |
| 3 | 64 | 21 |

Marking scheme: n = 1 (0.5+0.5), n=2 (0.5+0.5), n=3 (0.5+0.5)

(b) (7 points) Given an integer $n$ ($n > 0$), how many processes are executed, and how many lines of `fork() == fork()` are printed in the end? Please explain your solution. You should write your answer in terms of $n$:

Total number of processes are executed: _____

Explanation for the total number of processes:

Total number of lines of fork()==fork(): _____

Explanation for the total number of lines:

**Answer**:

Explanation Part 1: Total number of processes
(answer: 2 marks, explanation: 2 marks)

In each loop iteration, the parent process creates 2 new child processes, one of which creates another child (grandchild). In total, there will be 4 processes running at the end of an iteration.

In the end, there are **4^n** processes

Explanation Part 2: Total number of lines printed
(answer: 2 marks, explanation: 1 marks)

In each loop iteration, only the grandchild process sees the return values of the two fork() being equal (both returning 0). So only one message is printed. The pattern looks like the following:

When n = 0, 4 processes in the current iteration, 4/4 lines = 1 line is printed
When n = 1, 16 processes in the current iteration, 16/4 lines = 4 lines are printed
…
In general, when the current iteration is n, there are 4^n processes, 4^n/4 lines will be printed in that iteration.

In total, there are **4^0 + 4^1 + … + 4^(n-1)** messages printed out.

## Problem 3 CPU Scheduling [36 Points]

1) (12 points) The following table contains 5 processes (P1-P5) with its arrival time and burst time.

| Process | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0 | 6 |
| $P_2$ | 4 | 10 |
| $P_3$ | 10 | 12 |
| $P_4$ | 16 | 8 |
| $P_5$ | 20 | 2 |

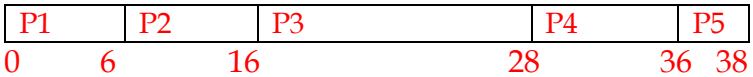
Whenever there is a tie among processes, e.g., the same arrival time, remaining time, and etc., they are inserted into the ready queue in the ascending order of process id.

For each of the following scheduling algorithms, draw the Gantt charts depicting the sequence of the process execution, and calculate the average turnaround time and average waiting time.

1. FCFS
2. RR (quantum=6)
3. SJF (non-preemptive)
4. SRTF

**Answer**:

- **FCFS** (3 points)



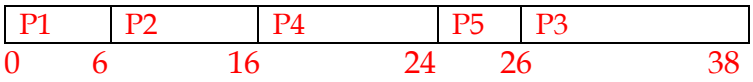

Avg turnaround: (6 + 12 + 18 + 20 + 18)/5 = 74/5 = 14.8
Avg waiting: (0 + 2 + 6 + 12 + 16)/5 = 36/5 = 7.2

- **RR** (3 points)

0 6 12 18 22 28 34 36 38

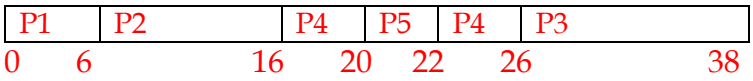
Avg turnaround: (6 + 18 + 24 + 22 + 16)/5 = 86/5 = 17.2
Avg waiting: (0 + 8 + 12 + 14 + 14)/5 = 48/5 = 9.6

- **SJF** (3 points)




Avg turnaround: (6 + 12 + 28 + 8 + 6)/5 = 60/5 = 12
Avg waiting: (0 + 2 + 16 + 0 + 4)/5 = 22/5 = 4.4

- **SRTF** (3 points)

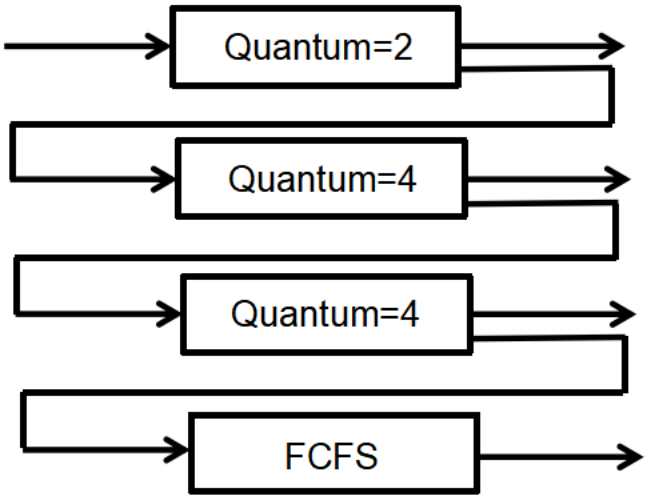



Avg turnaround: (6 + 12 + 28 + 10 + 2)/5 = 58/5 = 11.6
Avg waiting: (0 + 2 + 16 + 2 + 0)/5 = 20/5 = 4

2) (12 points) Consider the following single-thread processes, arrival times, burst times and the following four queues:

Q0 – RR with time quantum 2 milliseconds
Q1 – RR with time quantum 4 milliseconds
Q2 – RR with time quantum 4 milliseconds
Q3 – FCFS

| Process | Arrival Time | Burst Time |
|---|---|---|
| $P_1$ | 0 | 6 |
| $P_2$ | 1 | 4 |
| $P_3$ | 6 | 16 |
| $P_4$ | 9 | 5 |
| $P_5$ | 13 | 3 |
| $P_6$ | 20 | 12 |




a) (6 points) Draw the Gantt chart depicting the scheduling procedures for these processes.

**Answer**:

| P1 | P2 | P1 | P3 | P1 | P4 | P1 | P2 | P5 | P2 | P3 | P6 | P4 | P5 | P6 | P3 | P6 | P3 | P6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 4 | 6 | 8 | 9 | 11 | 12 | 13 | 15 | 16 | 20 | 22 | 25 | 26 | 30 | 34 | 38 | 44 | 46 |

b) (6 points) Calculate the average waiting time.

**Answer**:
P1: (4-2)+(8-6)+(11-9)=6
P2: (2-1) + (12-4) + (15-13) = 11
P3: (16-8) + (30-20) + (38-34) = 22
P4: 22-11 = 11
P5: 25-15 = 10
P6: (26-22)+(34-30)+(44-38) = 14
average waiting time = (6+11+22+11+10+14)/6=12.33

| Grading Criterion | | |
|---|---|---|
| 3.2 a | Understanding the basics, with several mistakes | 2 |
| | Almost correct, with one minor mistake | 4 |
| | correct | 6 |
| 3.2 b | Understanding the basics, with several mistakes | 2 |
| | Almost correct, with one minor mistake | 4 |
| | correct | 6 |

3) (12 points) Consider the following four single-thread processes (P1 - P4) with the executing times, deadlines and periods shown in the table. Assume all processes arrive at timeslot 0. Fill in the table with the ID of the process that is running on

the CPU with Rate-Monotonic (RM) scheduling and Earliest Deadline First (EDF) scheduling in the first 20 timeslots, and show how many deadlines are missed under each scheduler.

| Process | Processing Time | Deadline | Period |
|---|---|---|---|
| $P_1$ | 1 | 2 | 4 |
| $P_2$ | 1 | 5 | 6 |
| $P_3$ | 3 | 8 | 8 |
| $P_4$ | 2 | 7 | 15 |

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | | | | | | | | | | | | | | | | | | | | |
| EDF | | | | | | | | | | | | | | | | | | | | |

Answer:

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RM | 1 | 2 | 3 | 3 | 1 | 3 | 2 | 4 | 1 | 3 | 3 | 3 | 1 | 2 | 4 | 4 | 1 | 3 | 2 | 3 |
| EDF | 1 | 2 | 4 | 4 | 1 | 3 | 3 | 3 | 1 | 2 | 3 | 3 | 1 | 3 | 2 | 4 | 1 | 4 | 2 | 3 |

RM: 1 (P4), EDF: 0 (2 marks). Gantt charts with arrows (like lecture examples) will also be accepted.

RM(4 marks): [0, 5], [5, 10], [10, 15], [15, 20] 1 mark per interval
EDF(6 marks): [0, 4], [4, 7], [7, 10], [10, 15], [15, 18], [18, 20] 1 mark per interval

## Problem 4 Synchronization [14 Points]

1) (4 points) The program below is made up of two concurrent processes sharing semaphores X and Y, both initialized to 0. What will be the output when both processes complete execution? If there is more than one possible output, describe all possibilities.

```
/* Both X and Y are initialized to 0 */
```

| Process A | Process B |
|---|---|

| | |
|---|---|
| `wait(X);`<br>`cout << "b";`<br>`signal(Y);`<br>`wait(Y);`<br>`cout << "c";`<br>`signal(Y);` | `signal(X);`<br>`cout << "a";`<br>`wait(Y);`<br>`cout << "d";` |

Answers: `abcd, bacd, bcad` (3 possible answers)
Detailed marking scheme:
2 points for one correct output
3 points for two correct outputs
4 points for all correct outputs
One point will be deducted if the student writes incorrect outputs (no matter how many)

2) (10 points) Suppose that two threads execute the following C code concurrently, accessing *shared variables* **a**, **b**, and **c**:

**Initialization**
```
int a = -7;
int b = 2;
int c = 3;
```

| **Thread 1** | **Thread 2** |
|---|---|
| `if (a > 0) {`<br>`  c = b - a;`<br>`} else {`<br>`  c = b + a;`<br>`}` | `b = 8;`<br>`a = 4;` |

What are all the possible values for **c** after both threads complete? You can assume that reads and writes of the variables are *atomic*, and that the order of execution of statements within each thread is preserved by the C compiler and the hardware so it matches the code above.

**Answer: 4, 12, -5, 1, 6**
Case-1: the program executes the `a>0` branch, then we have `c = 8 - 4 = 4`
Case-2: the program executes the "else" branch. We have the following sub-cases:
a) Thread 2 finishes before the assignment of `c`, i.e., `c = 8 + 4 = 12`.
b) Thread 2 starts after the assignment of `c`, i.e., `c = 2 + (-7) = -5`
c) The assignment of `b` executes first; the assignment of `c` next, followed by the assignment of `a`, i.e., `c = 8 + (-7) = 1`.
d) The assignment of `c` executes first. After `b` is read, thread 2 executes, followed by `b+a`, i.e., `c = 2 + 4 = 6`.

Detailed marking scheme:
2 points for each correct output.
One point will be deducted if the student writes incorrect outputs (no matter how many)