# System and Device Programming
# Standard Exam
# 11.07.2024

## Ex 1 (1.25 points)

Analyze the following code snippet in C++. Indicate all possible admissible outputs. Note that wrong answers imply a penalty in the final score.

```
template<typename Container, typename Compare>
void sc (Container& container, Compare comp) {
  container.sort(comp);
}

int main() {
  std::list<std::string> list = {
    "apple", "orange", "banana", "grape", "pineapple"};
  auto dor = [](const std::string& a, const std::string& b) {
    return a > b;
  };
  sc (list, dor);
  for (const auto & str : list) {
    std::cout << str << " ";
  }
  return 0;
}
```

Choose one or more options:

1. ⬜ apple banana grape orange pineapple
2. ⬜ 5 6 5 6 9
3. ⬜ 9 6 5 6 5
4. ⬜ apple orange banana grape pineapple
5. ⬜ pineapple orange grape banana apple
6. ⬜ pineapple grape banana orange apple
7. ⬜ 5 6 6 5 9
8. ⬜ 9 5 6 6 5

## Ex 2 (1.25 points)

Analyze the following code snippet in C++. Indicate all possible admissible outputs. Note that wrong answers imply a penalty in the final score.

```
int main() {
  std::map<std::string, std::set<int>> myMap;
  myMap["third"].insert(6);
  myMap["third"].insert(8);
  myMap["third"].insert(7);
  myMap["second"].insert(4);
  myMap["second"].insert(5);
  myMap["first"].insert(2);
  myMap["first"].insert(1);
  myMap["first"].insert(3);
  for (const auto& pair : myMap) {
    std::cout << pair.first << ": ";
    for (const auto& num: pair.second) {
```

```
      std::cout << num << " ";
    }
  }
  std::string key = "second";
  if (myMap.find(key) != myMap.end()) {
    for (const auto& num : myMap[key]) {
      std::cout << num << " ";
    }
    std::cout << std::endl;
  } else {
    std::cout << "Key \"" << key << "\" not found." << std::endl;
  }
   return 0;
}
```

Choose one or more options:

1. ⬜ 1 2 3 4 5 6 7 8 4 5
2. ⬜ first: second: third:
3. ⬜ first: 1 2 3 second: 4 5 third: 6 7 8
4. ⬜ first:  second: third: 4 5
5. ⬜ first: 1 2 3 second: 4 5 third: 6 7 8 4 5
6. ⬜ first: 2 1 3 second: 4 5 third: 6 8 7 4 5
7. ⬜ third: 6 7 8 second: 4 5 first: 1 2 3 4 5
8. ⬜ first: 2 1 3 second: 4 5 third: 6 8 7 4 5

## Ex 3 (1.0 points)

Analyze the following code snippet reporting the use of a condition variable implemented in the library Pthread. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

```
pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
int sharedData = 0;
int ready = 0;

void* producer(void* arg) {
  pthread_mutex_lock(&mutex);
  sleep(1);
  sharedData = 42;
  ready = 1;
  pthread_cond_signal(&cond);
  pthread_mutex_unlock(&mutex);
  return NULL;
}

void* consumer(void* arg) {
  pthread_mutex_lock(&mutex);
  while (ready == 0) {
    pthread_cond_wait(&cond, &mutex);
  }
  printf("Consumer: consumed data = %d\n", sharedData);
  pthread_mutex_unlock(&mutex);
  return NULL;
}
```

```
int main() {
  pthread_t p, c;
  pthread_create(&p, NULL, producer, NULL);
  pthread_create(&c, NULL, consumer, NULL);
  pthread_join(p, NULL);
  pthread_join(c, NULL);
  pthread_mutex_destroy(&mutex);
  pthread_cond_destroy(&cond);
  return 0;
}
```

Choose one or more options:

1. ⬜ The `pthread_cond_wait` function in the consumer thread waits for the condition variable to be signaled, and, in the meantime, it releases the mutex.
2. ⬜ If `pthread_cond_signal` is called before `pthread_cond_wait`, the consumer thread will immediately proceed after calling `pthread_cond_wait`.
3. ⬜ If `pthread_cond_signal` is called before `pthread_cond_wait`, the signal will be lost, and the consumer thread may wait indefinitely.
4. ⬜ If `pthread_mutex_unlock(&mutex)` is not called in the producer thread after `pthread_cond_signal`, the consumer thread might be blocked indefinitely.
5. ⬜ If `pthread_mutex_unlock(&mutex)` is not called in the producer thread after `pthread_cond_signal`, the program would produce a compile-time error.
6. ⬜ The `while(ready==0)` in the consumer thread can be substituted by `if(ready==0)`.
7. ⬜ If more producers are run by the main, the condition variable synchronization does not work.
8. ⬜ If more consumers are run by the main, the condition variable synchronization works only if `pthread_cond_signal` is substituted by `pthread_condition_broadcast`.
9. ⬜ The shared variable `sharedData` is protected by a mutex to avoid race conditions.

## Ex 4 (3.5 points)

One C Unix process named P forks a child process C. P and C processes share a chunk of memory to exchange information and use a pipe to synchronize their effort. They behave as follows:

- P starts first.
    1. It generates a random number of float values and writes them on the shared memory.
    2. Then, it sends a token on the pipe to wake up C, who waits on the pipe.
    3. Finally, it waits for a token on the same pipe, and when the token is received, it restarts from step 1.
- C starts after P.
    1. It waits for a token on the pipe.
    2. Once it has received the token, it displays the floats stored in the shared memory on standard output, and it sends a token to P to state it has done.
    3. It iterates, resparting from step 1.

Notice that the number of float values stored in the shared memory is variable (but must be stored in the shared memory); thus, it can be communicated from P to C using the pipe. Implement in C code either the parent process P or the child process C, i.e., it is not requested to implement both of them.

## Ex 5 (2.0 points)

Explain how the dynamic memory allocation is managed in C++ code using standard and smart pointers. More specifically, illustrates the different types of pointers and allocation strategies. Describe the two approaches' main characteristics, advantages, and disadvantages.

## Ex 6 (3.5 points)

Write a program in C++ implementing a producer and consumer scheme using promises and futures. Two producers generate real values at random intervals (ranging from 1 to 5 seconds). When both values are ready, two consumers get those values, and then the first one displays the sum of those values, and the second one

shows their product. Once the consumers have done so, the entire cycle **in the main function** restarts (with the producers producing two values, etc.). The program ends when the sum or product of the two values (computed by the consumers) equals zero.

## Ex 7 (2.5 points)

A class C includes as a private attribute a pointer to an array of integer values. Write the class, defining the content of the constructor, copy constructor, copy assignment operator, move constructor, move assignment operator, and destructor. Define a small main which defines an object of such a type and activates the various copy constructors.