System and Device Programming
On Off Exam
22.01.2024

Ex 1 (1.5 points)

Suppose to run the following program using the command:
`./pgrm 2`
Indicate the possible admissible outputs. Note that wrong answers imply a penalty in the final score.

```
#define N 100

int main (int argc, char **argv) {
  char str[N];
  int n;

  setbuf (stdout, 0);
  printf("M");
  sscanf (argv[1], "%d", &n);
  if (n>0) {
    if (fork () > 0) {
      printf("E");
      sprintf (str, "%d", n-1);
      execlp (argv[0], argv[0], str, NULL);
    } else {
      printf("S");
      sprintf (str, "%s %d", argv[0], n-1);
      system (str);
    }
  }
  return 1;
}
```

Choose one or more options:

1. ✅ MEMEMSMSMEMSM
2. ⬜ MMEEMSMSMEMSM
3. ✅ MSMSMEMEMEMSM
4. ✅ MEMEMSSMEMMSM
5. ⬜ MESMMMESESMMM
6. ✅ MEMEMSMSEMMSM
7. ✅ MESMMESESMMMM

Ex 2 (1.5 points)

Suppose to run the following program. Indicate the possible admissible outputs. Note that wrong answers imply a penalty in the final score.

```
sem_t s1, s2, me1, me2;

static void *TA ();
static void *TB ();
static void *TC ();

int main (int argc, char **argv) {
  pthread_t th;
```

```
  sem_init (&s1, 0, 1);
  sem_init (&s2, 0, 0);
  sem_init (&me1, 0, 1);
  sem_init (&me2, 0, 0);

  setbuf(stdout, 0);

  pthread_create (&th, NULL, TA, NULL);
  pthread_create (&th, NULL, TB, NULL);
  pthread_create (&th, NULL, TC, NULL);

  pthread_exit(0);
}

static void *TA () {
  pthread_detach (pthread_self ());

  while (1) {
    sem_wait (&me1);
    for (int i=0; i<2; i++) {
      sem_wait (&s1);
      printf ("A"); fflush (stdout);
      sem_post (&s1);
      if (i==1) {
        sem_post (&me1);
      }
    }
    sem_post (&s2);
    sem_wait (&me2);
  }

  return 0;
}


static void *TB () {
  pthread_detach (pthread_self ());

  while (1) {
    sem_wait (&me1);
    for (int i=0; i<2; i++) {
      sem_wait (&s1);
      printf ("B"); fflush (stdout);
      sem_post (&s1);
      if (i==1) {
        sem_post (&me1);
      }
    }
    sem_post (&s2);
    sem_wait (&me2);
  }

  return 0;
}
```

```
static void *TC () {
  pthread_detach (pthread_self ());

  while (1) {
    sem_wait (&s2);
    sem_wait (&s2);
    printf ("C\n");
    sem_post (&me2);
    sem_post (&me2);
  }

  return 0;
}
```

Choose one or more options.

1. ⬜ The program repeatedly displays the following sequence followed by a new line: ABABC
2. ✅ The program repeatedly displays the following sequence followed by a new line: AABBC
3. ⬜ The program repeatedly displays the following sequence followed by a new line: AAAAC
4. ✅ The program repeatedly displays the following sequence followed by a new line: BBAAC
5. ⬜ The program repeatedly displays the following sequence followed by a new line: BBBBC
6. ⬜ The program repeatedly displays the following sequence followed by a new line: BABAC

Ex 3 (1.5 points)

Analyze the following code snippet in C++. Indicate the possible output or outputs obtained by executing the program.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

const int N = 3;

int main() {
  vector<string> v = {"this", "is", "a", "c++", "vector", "of", "strings", "!!!"};
  int n = count_if(v.begin(), v.end(), [](string s) {
    return (s.length()>N);
  });
  cout << n;
  return 0;
}
```

Choose one or more options:

1. ⬜ The program displays the value 4.
2. ⬜ The program displays the value 2.
3. ⬜ The program does not run as it contains a bug.
4. ✅ The program displays the value 3.
5. ⬜ The variable `s` in the lambda function is not defined.
6. ⬜ We must define the lambda function before the main program.

Ex 4 (1.5 points)

Analyze the following code snippet in C++. When the main is executed, indicate how many (standard) constructors, copy assignment operators, and destructors are called.

```
class C {
   private:
      ...
   public:
     ...
};

void f1(C e) { ... }
void f2(C &e) { ... }

int main() {
  C e1, e2;
  e2 = e1;
  C e3 = *new C;
  e2 = e3;
  return 0;
}
```

Solution

```
{1}[C][C]{2}[CAO]{3}[C][CC]{4}[CAO]{6}[D][D][D]
```

Choose one or more options:

1. ✅ 3 constructors, 2 copy assignment operators, and 3 destructors.
2. ⬜ 3 constructors, 3 copy assignment operators, and 3 destructors.
3. ⬜ 3 constructors, 3 copy assignment operators, and 4 destructors.
4. ⬜ 2 constructors, 3 copy assignment operators, and 3 destructors.
5. ⬜ 2 constructors, 2 copy assignment operators, and 3 destructors.
6. ⬜ 3 constructors, 2 copy assignment operators, and 4 destructors.

Ex 5 (1.5 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

```
#include <iostream>
#include <string>
#include <list>

using namespace std;

class myc {
  public:
    int rn;
    string s;

  myc (int lrn, string ls) {
    rn = lrn;
    s = ls;
  }
};

int main() {
  list<myc> l;

  cout << l.size() << " ";
  l.push_back(myc(10, "a"));
```

```
  l.push_back(myc(20, "bb"));
  l.emplace_back(30, "ccc");
  cout << l.size() << " ";
  while (!l.empty()) {
    cout << l.front().rn << " " << l.front().s << " ";
    l.pop_front();
  }

  return 0;
}
```

Choose one or more options:

1. ✅ The program displays the following sequence: 0 3 10 a 20 bb 30 ccc
2. ⬜ The program displays the following sequence: 0 3 a 10 bb 20 ccc 30
3. ⬜ The program displays the following sequence: 10 a 20 bb 30 ccc q
4. ⬜ The program displays the following sequence: a 10 bb 20 ccc 30
5. ⬜ The program displays the following sequence: 0 3 30 ccc 20 bb 10 a
6. ⬜ The program displays the following sequence: 0 3 ccc 30 bb 20 a 10

Ex 6 (1.5 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

```
#include <iostream>
#include <map>
#include <string>

using namespace std;

int main() {
  map<string, int> mp;

  mp["one"] = 1;
  mp["two"] = 2;
  mp["three"] = 3;
  map<string, int>::iterator it = mp.begin();
  while (it != mp.end()) {
    cout << it->first << " " << it->second << " ";
    ++it;
  }

  return 0;
}
```

Choose one or more options:

1. ✅ The program displays the following sequence: one 1 three 3 two 2
2. ⬜ The program displays the following sequence: one 1 two 2 three 3
3. ⬜ The program displays the following sequence: 1 one 3 three 2 two
4. ⬜ The program displays the following sequence: 1 one 2 two 3 three
5. ⬜ The program displays the following sequence: three 3 twp 2 one 1
6. ⬜ The program displays the following sequence: 3 three 2 two 1 one
7. ⬜ The program displays a sequence different from all the reported ones.

Ex 7 (1.0 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

Code in Thread 1

```
pthread_mutex_lock (&m1);
done = 1;
pthread_cond_signal (&cv);
pthread_mutex_unlock (&m1);
```

Code in Thread 2

```
pthread_mutex_lock (&m2);
pthread_cond_wait (&cv, &m2);
pthread_mutex_unlock(&m2);
```

Choose one or more options:

1. ✅ The function `pthread_cond_signal` can be substiturted by `pthread_cond_broadcast`.
2. ✅ Thread 1 and Thread 2 must use the same mutex, i.e., `m1` and `m2` must be the same mutex.
3. ⬜ In Thread 2, `pthread_cond_wait` must be inserted in an `if` construct.
4. ✅ In Thread 2, `pthread_cond_wait` must be inserted in a `while` construct.
5. ✅ If Thread 2 runs before Thread 1, Thread 2 will stop on function `pthread_cond_wait`, and it will release the mutex `m2`.
6. ✅ If Thread 1 runs before Thread 2, Thread 2 will stop on `pthread_cond_wait`.
7. ⬜ If Thread 1 runs before Thread 2, Thread 2 will not stop on `pthread_cond_wait`, because the `pthead_cond_signal` on the variable `cv` is lost.

Ex 8 (1.0 points)

Considering C++ programming with threads or tasks, promises, and futures, indicate which of the following statements is correct. Note that wrong answers imply a penalty in the final score.

Choose one or more options:

1. ✅ The number of threads may be higher than the number of hardware threads (over-subscription).
2. ⬜ The number of tasks may be higher than the number of hardware threads (over-subscription).
3. ✅ Threads in C++ do not offer a specific way to return a value to the caller.
4. ⬜ Tasks in C++ do not offer a specific way to return a value to the caller.
5. ⬜ Each thread has an asynchronous policy associated with it; the policy can be `launch::asynch`, `launch::deferred` or the default one.
6. ✅ A future is an object representing a value generated by some provider.
7. ⬜ A promise is an object that can represent a value generated by some provider.
8. ✅ A promise is an object that can store a value to be retrieved by a future object.
9. ⬜ Promises are stored in the producer of the promise.

Ex 9 (1.0 points)

Indicate which statements are correct. Note that wrong answers imply a penalty in the final score.

Choose one or more options:

1. ✅ An ASCII file is generally more compact than a UNICODE one.
2. ✅ Both an ASCII file and a BINARY file can be read with function `read`.
3. ⬜ Both an ASCII file and a BINARY file can be manipulated with functions `fscanf` and `fprintf`.
4. ⬜ An ASCII file is generally more compact than a BINARY one.
5. ⬜ Functions `fopen` and `open` return the same type of object.

6. ⬜ In a binary file, it is impossible to store integer values.
7. ✅ UNICODE characters can occupy 1, 2, or 4 bytes.
8. ✅ A BINARY file is generally more compact than an ASCII one.

Ex 10 (1.0 points)

Analyze the following code snippet. Indicate which of the following statements are correct. Note that wrong answers imply a penalty in the final score.

```
pthread_mutex_lock (&mutex);
count++;
if (count == N_THREAD) {
  sem_post (&sem);
}
pthread_mutex_unlock (&mutex);
sem_wait (&sem);
sem_post (&sem);
```

Choose one or more options:

1. ✅ The mutex named mutex can be substituted by a second semaphore sem2 initialized to 1.
2. ⬜ The piece of code can be used exactly as it is (as a unique synchronization strategy) by a thread that loops through a cycle and hits the barrier at every iteration.
3. ✅ The piece of code includes a barrier implemented in a correct way.
4. ⬜ The variable count must be initialized to 0 by each thread.
5. ✅ The semaphore named sem must be initialized to 1.
6. ⬜ The piece of code includes a barrier implemented in a wrong way, as it is necessary to insert a cycle to free all waiting threads.

Ex 11 (1.0 points)

Modern operating systems provide memory-mapped files to associate a process's address space with a file. Indicate which of the following statements are correct. Note that more than one response can be correct and that incorrect answers may imply a penalty on the final score.

Choose one or more options:

1. ✅ Memory mapping allows the programmer to manipulate files without file I/O functions.
2. ✅ A memory-mapped region is unmapped when the process terminates, or we call function unmap.
3. ⬜ Function `munmap` forces the contents of the mapped region to be written to the disk file.
4. ⬜ We can use the function `mmap` to map the file to memory and to resize it.
5. ⬜ With read/write, we manipulate the data directly, whereas with mmap we copy the data through the kernel.
6. ⬜ Once a file has been mapped, it can be manipulated only through function memcpy.

Ex 12 (1.0 points)

Concerning inter-process communication, indicate which of the following statements is correct. Note that more than one response can be correct and that incorrect answers may imply a penalty on the final score.

Choose one or more options:

1. ⬜ FIFO can only be used between processes sharing an ancestor.
2. ✅ Function `ftok` shares a key among different processes.
3. ✅ A message queue is a linked list of messages stored within the kernel.
4. ⬜ FIFO allows only blocking operations.

5. ⬜ A FIFO differs from a pipe because it includes a linked list of messages stored within the kernel.
6. ✅ Pipes can be used only between processes sharing an ancestor.
7. ⬜ Function `msgctl` sends a message in a message queue.
8. ✅ FIFOs are used to pass streams of anonymous bytes.