CS31: Midterm – Solution

CS31, Winter 2013

Feb 6th, 2013

Advices and guideline

- No partial credit will be given for multiple-choices question. The answer is either correct or incorrect.
- There will be up to 25% of the question points removed for wrong answers, except for problem 1.8. Do not guess!
- Cheating, in any form, is forbidden and will be reported after the exam.
- Sharing notes and/or information with other student(s), in any form, is forbidden and will be reported after the exam.

1 Problem 1

In this problem, you manipulate a small library to schedule tasks by order of precedence. This is typically used in resource planning, where you have several tasks required to perform a complete process, and must order them effectively. Think for instance about the process of building a car, with four tasks: (A) build the engine; (B) build the frame; (C) put the engine in the car; (D) paint the car. In this example, the four tasks can be organized such that we do first (A) and (B) at the same time (they do not depend on each other), and then only when both are completed we can do (C), and then when (C) has completed we can do (D).

The following is a code sample for a scheduler where exactly four arbirary tasks can be scheduled. We have removed all comments, and possibly introduced errors in the code. You will have to carefully read the code, understand it and where the problems are, if any, and answer questions. **No knowledge or understanding about the general problem of task scheduling is required to do this exercise**. Reading and understanding C++ is the only thing you need.

```
#include <iostream>

namespace myScheduler {

  typedef const char task_id;
  typedef int task_time;
  typedef const int error_code;

  task_id T1 = 'A';
  task_id T2 = 'B';
  task_id T3 = 'C';
  task_id T4 = 'D';
  task_time t1, t2, t3, t4;
  error_code TIME_ERROR = -1;

  void set_task_time(task_id T, task_time val)
  {
    switch (T)
      {
      case T1: t1 = val; break;
      case T2: t2 = val; break;
      case T3: t3 = val; break;
      case T4: t4 = val; break;
      }
  }

  int get_time(task_id T)
  {
    switch (T)
      {
      case T1: return t1;
      case T2: return t2;
      case T3: return t3;
      case T4: return t4;
      default: return TIME_ERROR;
      }
  }

  void print_schedule()
  {
    std::cout << "Schedule:("
              << T1 << "=" << t1 << ", "
              << T2 << "=" << t2 << ", "
              << T3 << "=" << t3 << ", "
              << T4 << "=" << t4 <<  ")" << std::endl;
  }

  void schedule(std::string pred)
  {
    unsigned int i;
    bool has_converged;
    do
      {
        has_converged = true;
        for (i = 0; i < pred.size(); i += 2)
          {
            task_time val1 = get_time(pred[i]);
            task_time val2 = get_time(pred[i+1]);
            if (val1 >= val2)
              has_converged = false;
            task_time max = val1 + 1;
            if (max < val2)
              max = val2;
            set_task_time(pred[i+1], max);
          }
      }
    while (! has_converged);
  }

  void reset_tasks()
  {
    t1 = t2 = t3 = t4 = 0;
  }


```

```
76
77    int min_time()
78    {
79      int min = t1;
80      if (t2 < min) min = t2;
81      if (t3 < min) min = t3;
82      if (t4 < min) min = t4;
83      return min;
84    }
85
86    int assign_result(int& a, int& b, int& c, int& d)
87    {
88      a = t1; b = t2; c = t3; d = t4;
89      return min_time();
90    }
91
92  }
93
94  int main()
95  {
96    using namespace myScheduler;
97
98    reset_tasks();
99    schedule("ABACBC");
100   print_schedule();
101
102   return 0;
103 }
```

Question 1 (5 points) We focus on line 14. Answer the following question (multiple answers possible):

None of the proposed answers is correct.

Question 2 (5 points) We now consider the full program. Answer the following question (multiple answers possible):

None of the proposed answers is correct: the code compiles if using G++.

There is at least one another compilation error, not described in the other answers: missing `#include <string>`) if using VC++.

This question has been cancelled. This midterm is now graded on a maximal score of 95.

Question 3 (10 points) We now focus particularly on the behavior of the function `schedule(std::string pred)`. We assume all fixes needed (if any) have been made to make this code compile. We replace lines 98 to 100 in the program by the following sequence of instructions:

`reset_tasks(); schedule("ABACBDCDAD"); print_schedule();`

What is printed when running the program?

`Schedule:(A=0, B=1, C=1, D=2)`

Question 4 (10 points) We still focus particularly on the function `schedule(std::string pred)`. We assume all fixes needed (if any) have been made to make this code compile. We replace lines 98 to 100 in the program by the following sequence of instructions:

`reset_tasks(); schedule("abacbdcdad"); print_schedule();`

What happens when running the program?

The program is stuck in an infinite loop.

Question 5 (5 points) We assume all fixes needed (if any) have been made to make this code compile. We replace lines 98 to 100 in the program by the following sequence of instructions:

```
int a, b, c, d;
reset_tasks();
schedule("BACD");
assign_result(a, b, c, d);
```

After the complete execution of all those instructions, which of the following is true (multiple answers possible)?:

The value of `a` is equal to `t1`

Question 6 (5 points) The algorithm implemented by the function `schedule(std::string pred)` does not operate properly for a collection of cases. Which of the following inputs are examples of problematic cases, that is, the program either crash, loops infinitely, or has an undetermined behavior (multiple answers possible)?

`"ABBAABBA"`

`"abababab"`

`"abbaabba"`

Question 7 (10 points) We assume all fixes needed (if any) have been made to make this code compile. We make a subsequent modification: we erase lines 59 and 60 from the program (that is the if conditional). Give a possible value for `my_answer`, the argument of `schedule` such that the output produced by the following sequence:

```
reset_tasks(); schedule(my_answer); print_schedule();
```

differs from the output produced by the original program (that is, a version that compiles and contains lines 59 and 60). In both cases (with and without lines 59 and 60), the function `schedule` must terminate without any problem (no crash / infinite loop / error message).

```
"ABADACDBCD"
```

Question 8 (25 points) For this question, you will extend the library `myScheduler` with a new function, that asserts the validity of a schedule. The context is precisely the same as for the code above: we have no more than four tasks, noted with the characters `A`, `B`, `C` and `D`. Each of them is associated an integer, `t1`, `t2`, `t3` and `t4`. Your function should plug seamlessly in the existing library, and you can call functions from `myScheduler` in your algorithm.

Write an algorithm (in algorithmic language, pseudo-code, or C++, whichever is most convenient for you) that implements the following specification: "A function that takes as argument a string `s` in identical format as the function `schedule`, and four integers t1, t2, t3, t4 representing the schedule of the four tasks A, B, C and D. This function returns true if (A=t1,B=t2,C=t3,D=t4) is a valid schedule for `s`, false in any other case."

We complement this specification by defining what is a valid schedule.

Given a string `s` which contains a set of pairs of characters, each pair represents a constraint on the scheduling order (a dependence) between the task associated with the first character and the task associated with the second character. A schedule for the task is valid if the time associated to a task is such that no task which is in dependence with it has a lower or equal time associated to it.

For instance, with the pair `"AB"`, the task `A` must have a time `t1` associated to it strictly lower than the time `t2` associated to the task `B`: `t1 < t2` must be a true expression in this case. The pair `"AB"` does not provide any constraint on `C` and `D`, that is `t3` and `t4` can take any value in this example. Another example: with the string `"ABBC"`, `B` depends on `A`, `C` depends on `B`, and so `C` depends on `A` too. `t1=0, t2=1, t3=2, t4=0` is a valid schedule for `"ABBC"`, but `t1=0, t2=2, t3=1, t4=0` is not.

To do correctly this exercise, you must:

- Have understood the encoding in the myScheduler library for the string `pred`, as you must use the same. No more information will be given to you.
- Correctly cover all cases of possible input strings and schedules. Points will be deduced for each case not covered. Your algorithm must work properly (no infinite loop, no crash, no error message) for ANY possible input string and ANY possible value for the four integers. That is, your function will be more robust in terms of correctly handled inputs than the proposed `schedule` function. When the input string is not well formed, your algorithm returns false (same as when the input string is well formed but the schedule given as input is not a valid schedule).
- Be careful about all cases of schedules that can exist: for instance, `t1=1,t2=2,t3=3,t4=4` is a valid schedule for `"ABACAD"`, but is not the schedule produced by the function `schedule("ABACAD")`: this function would produce `t1=0,t2=1,t3=1,t4=1`. Another perfectly valid schedule for the same string is `t1=-3,t2=-2,t3=-2,t4=-1`, for instance.

Possible algorithm:

Here, the algorithm is simple: take out special cases like odd number of letters, lower case letters, letters which are not A, B, C or D. Then scan once the string and ensure that gettime(letter1) ¡ gettime(letter2). Any algorithm more complex is trying to deal with cases covered already by this one: for instance transitive dependences as AC in ABBC are automatically covered.

```
Algorithm checkValidity
   Input:
        string s
        integer t1, t2, t3, t4
   Output:
        Boolean isValid

   for i in 0..s.size-1 do
        if s[i] != 'A' and s[i] != 'B' and s[i] != 'C' and s[i] != 'D'
             return false
        endif
        if s.size mod 2 != 0
             return false
        endif
   done
   for i in 0..s.size-1 step 2 do
        a := getTimeofLetter(s[i], t1, t2, t3, t4)
        b := getTimeofLetter(s[i + 1], t1, t2, t3, t4)
        if (a >= b)
             return false
        endif
   done
   return true

Algorithm getTimeofLetter
   Input:
        character c
        integer a, b, c, d
   Output:
        integer val
   if (c = 'A') then return a endif
   if (c = 'B') then return b endif
   if (c = 'C') then return c endif
   if (c = 'D') then return d endif
```

2 Problem 2 (25 points)

Question 1 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int foo() {
2    int a = 0;
3    return a;
4  }
5  int main() {
6     return foo();
7  }
```

`main` returns systematically 0.

Question 2 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int main() {
2     int a, b, c = 0;
3     c *= b * a + c - d;
4     return c;
5  }
```

The program does not compile.

Question 3 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int foo(int a, int b) {
2    return a;
3  }
4  int main() {
5     int foo = foo(1, 2);
6     return foo;
7  }
```

The program does not compile.

Question 4 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  namespace foo {
2   int foo(int a, int b) {
3     a = b;
4     return a;
5   }
6  }
7  int main() {
8     int foo = foo::foo(1, 2);
9     return foo;
10 }
```

`main` returns 2.

Question 5 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int main() {
2     int a = 0;
3     for (i = 0; i < 10; ++i)
4        a += i;
5     return i;
6  }
```

The program does not compile.

Question 6 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int main() {
2     int a = 0;
3     for (int a = 0; a++ < 10; ++a)
4        ;
5     return a;
6  }
```

None of the proposed answers is correct.

Question 7 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int main() {
2     float a, b, c, d;
3     a = 1; b = 2; c = 3; d = 4;
4     int d = a + b / c;
5     return d;
6  }
```

The program does not compile.

Question 8 (2 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  using namespace std;
2  int main() {
3     std::cout << "Hello World!" << std::endl << std::endl;
4     return 0;
5  }
```

The code produces one or more error(s) and/or warning(s)

Question 9 (3 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int bar(int a) {
2     foo(1, 2);
3     return 0;
4  }
5  void foo(int a, int b) {
6    a = b;
7  }
8  int main() {
9    int blob = bar(42);
10   return ++blob;
11 }
```

The program does not compile

Question 10 (3 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int main() {
2    int a = 0;
3    do {
4      for (a = 0; a < 10 && a > 10; ++a)
5        --a;
6    } while (a != 0);;
7    return a--;
8  }
```

`main` returns 0.

Question 11 (3 points) Study the following complete program (the entire `.cpp` file is shown), and answer the question below.

```
1  int main() {
2    int a = 0;
3    do {
4      switch (a) {
5        case 42:
6          for (a = 0; a < 10; ++a)
7          break;
8        default:
9          return 42;
10     }
11   } while (a != 0);
12   return -1;
13 }
```

`main` returns 42.