Exam — Informatik D-MATH/D-PHYS
24. 1. 2014 09:00–11:00

Prof. Bernd Gärtner
Examinee:
Last name: ...................................................................................................................
First name: ...................................................................................................................
Stud. no.: ...................................................................................................................

With my signature I attest, that I was able to sit the exam under regular conditions and that I read and understood the general remarks.

Signature: ....................................................................................................................

General remarks:

1. Verify the completeness of the exam given to you (four two-sided sheets with 6 assignments in total and one empty sheet for notes)! Fill in the title sheet, i.e., state your name and your student number, in a legible form.

2. Allowed utilities: None, except dictionaries.

3. Cheating or attempts to cheat lead to immediate disqualification and may entail legal consequences.

4. Write your solutions directly on the assignment sheets! Only one solution attempt per assignment will be considered. Hint: Prepare your solutions on separate sheets and transfer only the final solutions to the assignment sheet. Clearly cross out solution attempts that you do not want to be considered. If the space provided on the assignment sheets is insufficient for your solution attempts, use separate sheets and label them with the assignment numbers and your name.

5. If you want to submit early, hand all relevant documents over to one of the invigilators before leaving the room.

6. After 10:45, it is no longer possible to submit early. Please remain seated until the exam ends and until your documents have been collected by the invigilators.

7. The exam’s pass mark is 60 out of 120 points. Good luck!
Assignment 1. (18 Points) For each of the following 6 expressions, provide C++ type and value! No points will be awarded for intermediate steps of the evaluation. Assume that \( x \) has type int and value 3 \textit{at the beginning of each subtask}.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 0 &lt; 1 &amp;&amp; 1 != 1</td>
<td></td>
<td>++x &gt; 3 )</td>
</tr>
<tr>
<td>( 12.0 / 3u / 2u )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2 + x++ )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 1.4e3 * 0.2e-2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( 2014 % 4 + x % 2 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( x / 2 + 9.0f / 6 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Assignment 2. (15 Points) For each of the following three code fragments, write down the sequence of numbers that it outputs!

a) for (int i=0; i<100; i*=2)  
   std::cout << ++i << " ";

Output:

b) int i=0;
   int j=5;
   while (i != j)  
   std::cout << (i+=2) - j++ << " ";

Output:

c) void f(unsigned int x)  
   if (x == 0)  
      std::cout << "-";
   else  
      std::cout << "+";
   f(x-1);
   std::cout << "+";
   
Output:
Assignment 3. (23 Points) In the game of can knockdown, cans are stacked on top of each other like in a pyramid. Each can has a certain value. This value is stored in a two-dimensional array. The first index of the array describes the row, the second index the position (column) within this row. We count rows from top to bottom, starting at 0, and columns from left to right, starting at 0. In the example below, the can in row 2, at position 0 has value 4 (D[2][0] == 4).

If a can gets hit, it will fall. All cans that are standing directly above another falling can will fall too. For example, if the can with value 7 in the middle of row 2 gets hit, the cans with values 7, 2, 3, and 1 will fall.

```c
int D[4][4] = {{1, 0, 0, 0},
               {2, 3, 0, 0},
               {4, 7, 6, 0},
               {7, 3, 2, 10}};
```

The following is the skeleton for a function hit, that computes the sum of the values of all fallen cans, if can at position/column c in row r was hit. Complete the skeleton to a correct function by providing the four expressions expr1, expr2, expr3 and expr4! You can assume that you can access the array D directly within the function, that is, D is a global variable.

```c
// PRE: the array D has size at least (n+1) x (n+1) (n = max(r, c)),
// D[r][c] denotes the value of the can in row r, column c
// where c <= r
// POST: returns the sum of the values of all cans that fall
// if the can at position/column c in row r gets hit
int hit(int r, int c)
{
    if (r == 0)
        return expr1;
    if (c == 0)
        return expr2;
    if (c == r)
        return expr3;
    return expr4;
}
```
// PRE: the array D has size at least (n+1) x (n+1) (n = max(r, c)),
// D[r][c] denotes the value of the can in row r, column c
// where c <= r
// POST: returns the sum of the values of all cans that fall
//       if the can at position/column c in row r gets hit
int hit(int r, int c)
{
    if (r == 0)
        return expr1;
    if (c == 0)
        return expr2;
    if (c == r)
        return expr3;
    return expr4;
}

:expr1:

:expr2:

:expr3:

:expr4:
Assignment 4. (24 Points) Your task is to provide an implementation of a function that computes a floating-point approximation of the exponential function

\[ e^x := \sum_{i=0}^{\infty} \frac{x^i}{i!}, \]

for a given real number \( x \). The implementation should be based on this definition of \( e^x \) as an infinite series. To obtain an approximation of \( e^x \), the function should sum up initial terms of this series (in double precision floating-point arithmetic) until adding the current term does not change the value obtained so far anymore. You are not allowed to use any external libraries. The total number of statements in your solution must not be larger than the number of lines provided.

// POST: returns e^x
double exp (double x)
{
    // 1
    // 2
    // 3
    // 4
    // 5
    // 6
    // 7
    // 8
    // 9
    // 10
    // 11
}
a) Provide correct definitions of the three public member functions! The total number of statements in your solution must not be larger than the number of lines provided.

    image::image()
    {
        // 1
        // 2
        // 3
    }
void image::set_pixel (int r, int c)
{
    // 1
}

image image::operator! () const
{
    // 1
    // 2
    // 3
    // 4
    // 5
    // 6
}

b) Assuming that there is an output operator for the class image, provide a definition of the main function that outputs the rightmost image on the previous page!

int main()
{
    // 1
    // 2
    // 3
    // 4
    // 5
    // 6
    // 7
    // 8
}

Assignment 6. (10 /4 Points) Let \( n \) be a positive natural number. A board of size \( 2 \times n \) shall be filled with domino tiles of size \( 1 \times 2 \), where each domino tile looks like this:

![Domino Tile]

Let \( T(n) \) be the number of different possibilities of filling the board. We obtain for example \( T(1) = 2 \):

\[
\begin{array}{ll}
++ & \circ \\
\circ & 
\end{array}
\]

For \( n = 2 \) we get \( T(2) = 8 \):

\[
\begin{array}{cccc}
++ & + & + & + \\
\circ & \circ & \circ & \circ \\
++ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ \\
\end{array}
\]

For \( n = 3 \), we just show one of the many possibilities.

\[
\begin{array}{llll}
++ & + & + & + \\
\circ & \circ & \circ & \circ \\
++ & \circ & \circ & \circ \\
\circ & \circ & \circ & \circ \\
\end{array}
\]

a) Give a recursive formula for \( T(n) \) if \( n \geq 3 \). You do not have to prove that the formula is correct.

\[ T(n) = \quad , \text{ if } n \geq 3. \]

b) Compute, using a) or any other means, the value \( T(4) \).

\[ T(4) = \]