Question 1.
Pizzeria Alfredo sells pizzas of varying sizes and numbers of toppings. They would like to have a program that computes the selling price of a pizza. The pizza base costs £0.001 per cm\(^2\) and the cost for each topping are £0.002 per cm\(^2\). In order to make profit, they multiply the cost of a pizza by a factor of 1.5.

Write a function \texttt{alfredo} that takes as inputs the diameter (in cm) and the number of toppings of the pizza, both given as integers, and calculates the price of the pizza as a floating point number. Diameters must be between 10 and 100 cm and the numbers of toppings must be between 1 and 10. The program should reject inputs outside these ranges.

[20 marks]

Question 2. Modular exponentiation is defined by

\[
\text{expmod}(b, e, m) = b^e \mod m
\]

where \(b, e, m\) are integers with \(b > 0\), \(e \geq 0\) and \(m > 0\).

Remark: Modular exponentiation plays an important role in cryptography.

Give an efficient recursive definition of \texttt{expmod} using the following idea of \textit{repeated squaring}: Assume \(e\) is a positive even number. Setting \(p = \text{expmod}(b, e \div 2, m)\), one can express \(\text{expmod}(b, e, m)\) in terms of \(p^2\), \(m\) and \(\text{mod}\).

In order to find the solution you may use the fact that \texttt{mod} commutes with multiplication, that is,

\[(x \ast y) \mod m = ((x \mod m) \ast (y \mod m)) \mod m\]

If \(e\) is an odd number, then one can, with a similar calculation, reduce the computation of \(\text{expmod}(b, e, m)\) to the computation of \(\text{expmod}(b, e - 1, m)\).

Use your fast implementation of \texttt{expmod} to compute the last 6 digits (in decimal notation) of \(x^e\) where \(x\) is your student number. Compare your program with a naive implementation of \texttt{expmod}.

[40 marks]
**Question 3.** Consider the following functions:

```hs
type Pt = (Float, Float)

project :: Pt -> Pt
project (x, y) = (x, 0)

project1 :: Pt -> Pt
project1 p = (fst p, 0)
```

Both functions project a point onto the $x$-axis. Nevertheless they are not equivalent. This can be seen by evaluating the expressions \( \text{snd} (\text{project } \text{crazy}) \) and \( \text{snd} (\text{project1 } \text{crazy}) \) where

```hs
crazy :: Pt
crazy = crazy
```

Explain the different behaviours by analysing the respective reductions. \[20 \text{ marks}\]

**Question 4.** Write a higher order program that approximately computes the integral \( \int_a^b f \) where \( a, b \) are real numbers and \( f \) is a real function. Your function will need an extra parameter \( d \) (a real number) representing the distance of the sampling points.

Use the following (naive) idea to write a recursive program: If \( a \geq b \), then \( \int_a^b f = 0 \). Otherwise, \( \int_a^b f \) is approximated by \( f(a) \cdot d + \int_{a+d}^b f \).

Fix a small sampling distance \( d \), say \( d = 0.001 \), and compute the approximate integral of various functions (for example polynomials or trigonometric functions) with varying integration bounds \( a, b \). Compare the computed results with the exact results you know from school. \[20 \text{ marks}\]

**Instructions for solving and submitting coursework**

1. For your coursework use the template file available at

   http://www.cs.swan.ac.uk/~csulrich/fp1.html

2. Write comments on your solutions and for each of your programs at least two of your inputs and hugs' responses (using copy and paste) into the sections

   {- Comments/Tests:
   
   -}

3. Type the submission date and your name in your solution file, print it, attach a signed submission form (available at the students office) and put your coursework in the wooden box on the second floor.

4. Late submissions will be penalised by taking off marks.