EXERCISES C++
SHEET 1

1 Basic Data Types, Casting and Operators

Problem 1. Write a C++ program which prints five different basic data types and the number of bytes that each uses. Try out various combinations of **long** and **unsigned** etc.

Problem 2. Cast **char** values to **int** values to find codes used for the characters 'a', 'z', 'A', 'Z', '0', and '9'.

Problem 3. Write a program that reads in a character **char** from the keyboard and then displays one of the following messages:

- If <char> is a lower case letter, the message "The upper case character corresponding to <char> is ..."
- If <char> is an upper case letter, the message "The lower case character corresponding to <char> is ..."
- If <char> is not a letter, the message "<char> is not a letter"

Problem 4. Write a program where the user enters a decimal number and the code prints out the nearest integer. You should use cast as part of your solution. Write a second version using the **cmath** library.

Problem 5. The following code contains several bugs. Fix them.

```
# include <iostream>
# include <cmath>

using namespace std;

int main(){
   cout<<"Type 0 for stone, ";
   cout<<"1 for scissors, 2 for paper \n";
   cout<<"Enter player 1's move \n";
   cin>>player1;
   cout<<"Enter player 2's move \n";
   cin>>player2;
   if (player1=player2) {
      cout<<"Its a draw \n";
```

```
}  else {
     diff=player1-player2;
     if (diff==-2 || diff==1){
         cout<<"Player 1 won \n";
     }  else {
         cout<<"Player 2 won \n";
     }
   }
}
```

2 Flow of control and user-defined functions

Problem 1. Write three functions which compute the factorial of a natural number n (i.e. $n!$) using: a **for** loop, a **while** loop and a **do-while** loop.

Problem 2. Write a program which will raise any number x to a positive power n using **for** loop. Write a second version using the **cmath** library.

Problem 3. Write a recursive function to compute the sum of the numbers between 1 and n.

Problem 4. Write a function that takes two integer parameters a and b and prints out all the numbers from a to b.

Problem 5. The n-th Fibonacci number can be defined by $x_n = x_{n-1} + x_{n-2}$ if $n \geq 2$. We define $x_0 = 1$ and $x_1 = 1$. Write a function **fibonacci** that evaluates the n-th Fibonacci number by recursion.

Problem 6. Write a function which prints the square roots of the equation:

$$ax^2 + bx + c = 0.$$

Problem 7. A commonly occuring function in mathematics is the cumulative normal function defined by:

$$\textbf{normcdf}(x) = N(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} exp(-\frac{t^2}{2}).$$

If $(x \geq 0)$ we define:

$$k(x) = \frac{1}{(1 + 0.2316419x)}.$$

A **good approximation** for $N(x)$ is given by:

$$1 - \frac{1}{\sqrt{2\pi}} exp(-\frac{x^2}{2}) k(0.319381530 + k(-0.356563782 + k(1.781477937 + k(-1.821255978 + 1.330274429k))$$

For $x \leq 0$ you can use the same formula to evaluate $1 - N(-x)$.

The formula can be derived by choosing the general functional form and then finding the coefficients that give the best fit. For this question, you should just accept the formula on face value.

Write a function called **normcdf** to evaluate the cumulative normal function.

In order to simplify your **normcdf** function use a **Horner** function which is defined as follows:

For each $n \in \mathbb{N}$,

$$h_0(x, a_0) = a_0,$$

$$h_n(x, a_0, a_1, ..., a_n) = a_0 + x h_{n-1}(x, a_1, a_2, ..., a_n).$$

We call these "Horner functions" because they use the Horner method of evaluating a polynomial. Any polynomial in x can be written as

$$h_n(x, a_0, a_1, ..., a_n) = a_0 + x h_{n-1}(x, a_1, a_2, ..., a_n)$$

for appropriate constants a_i. The advantage of using h to evaluate the polynomial is that you don't have to compute high powers of x.

Problem 8. Implement the Moro algorithm for the inverse function of the cumulative normal distribution. Call the resulting function **norminv**.

Moro algorithm

Suppose $x \in [0, 1]$. Define y=x-0.5. If $|y| < 0.42$, define $r = y^2$ and approximate **norminv** with the following formula: $y \frac{h_3(r, a_0, a_1, a_2, a_3)}{h_4(r, 1.0, b_1, b_2, b_3, b_4)}$, with:

Suppose $|y| >= 0.42$. If y is negative let $r = x$. Otherwise let $r = 1 - x$. Define

$$s = (log(-log(r))).$$

a_0= 2.50662823884;
a_1= -18.61500062529;
a_2 = 41.39119773534;
a_3= -25.44106049637;
b_1= -8.47351093090;
b_2= 23.08336743743;
b_3= -21.06224101826;
b_4= 3.13082909833;
c_0= 0.3374754822726147;
c_1= 0.9761690190917186;
c_2= 0.1607979714918209;
c_3= 0.0276438810333863;
c_4= 0.0038405729373609;
c_5= 0.0003951896511919;
c_6= 0.0000321767881768;
c_7= 0.0000002888167364;
c_8= 0.0000003960315187;

Define t by

$$t = h_8(s, c_0, c_1, ..., c_8).$$

If $x > 0.5$, **norminv** is approximated by t, otherwise by $-t$.

3 Pointers

Problem 9. Declare **int** variables **x** and **y** and **int*** pointer variables **p** and **q**. Set x to 2, y to 8, p to the address of x, and q to the address of y. Then print the following information:

(1) The address of x and the value of x.

(2) The value of p and the value of $*p$.

(3) The address of y and the value of y.

(4) The value of q and the value of $*q$.

(5) The address of p.

(6) The address of q.

Problem 10. Consider the function **void add1(int *n)** which increments by 1 the value at which the pointer points to.