SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING
University of the Witwatersrand, Johannesburg
Software Development II

Class Test 2022: 1 Hour – 35 marks

Instructions

- Answer *all* questions. The questions do not carry equal weight.
- For questions which require you to write source code, note that:
  - You only need to specify `#include`'s if specifically asked.
  - For classes, you can give the implementation entirely in the header file, unless directed otherwise.
  - Marks are not awarded solely for functionality but also for good design, making appropriate use of library functions, following good coding practices, and using a modern, idiomatic C++ style.
  - Your code must be easily understandable or well commented.
  - You may use pencil but then you forfeit the right to query the marks.
- Reference sheets are provided separately.

Question 1

Pascal's triangle is a triangular array of the binomial coefficients. The formula for finding the nth coefficient of the rth row of the triangle is given by:

$$\frac{r!}{n!(r-n)!}$$

In this formula both r and n start from zero. $r = 0$ refers to the starting row at the top of the triangle; $n = 0$ refers to the leftmost coefficient in each row.

Listing 1 contains a C++ program for calculating Pascal's triangle. The output of this program, for a specific input, is shown in Listing 2.

```
using namespace std;

int fact(int n)
{
    return n > 1 ? fact(n - 1) * n : 1;
}

int main()
{
    int r, r_max, i, value;

    cout << "Enter the number of rows of Pascal’s Triangle\n";
    cin >> r_max;

    for(r = 0; r < r_max; r++)
    {
        // Print leading spaces
        for(i = r; i <= r_max; i++)
            cout << "  ";

        for(i = 0; i <= r; i++)
        {
            value = fact(r)/(fact(i)*fact(r-i));
            cout << "  " << value;
        }
        cout << endl;
    }

    return 0;
}
```

Listing 1: Code for calculating Pascal’s triangle

```
Enter the number of rows of Pascal’s Triangle
5
              1
            1  1
          1  2  1
        1  3  3  1
      1  4  6  4  1
```

Listing 2: The output given by Listing 1 for an input of 5

Your task is to *refactor* the program in Listing 1 by applying good coding principles and practices, and making use of modern C++. The refactored program must still produce *exactly the same output* as the original program for any given input.

[Total Marks 13]

Question 2

a) Both the `vector` container and the `list` container (a doubly-linked list) have a `push_back` function which appends an element to the end of the contained sequence of elements. Assume that both containers already contain the integers: 1, 2 and 3. Draw the "before-picture" of what *each* container looks like before a fourth element (the element 4) is added through `push_back`, and an "after-picture" showing what they look like after the element has been added via `push_back`. (4 marks)

b) How would you expect these two different containers to compare in terms of time efficiency when using `push_back`? (4 marks)

c) A `vector` container is said to offer *random access* to its elements. Explain what the term *random access* means and why this is possible with a `vector`. (3 marks)

[Total Marks 11]

Question 3

a) Write the code for the swap function that appears on line 6 in the test below (Listing 3) . This function must correctly *swap the two arguments* so that the test passes. You may not make use of any STL `swap` function in your solution.

```
 1  TEST_CASE("x and y should be swapped")
 2  {
 3      auto x = 5;
 4      auto y = 2;
 5
 6      swap(x, &y);
 7
 8      CHECK(x == 2);
 9      CHECK(y == 5);
10  }
```

Listing 3: Testing the swap function

(5 marks)

b) Is it possible for a function with the following signature to swap the arguments that are passed in by the caller? Explain your answer.

```
void swap(int x, int y);
```

(2 marks)

c) Will the code in Listing 4 work as expected? Explain your answer.

```cpp
using namespace std;

void assign(int* ptr, int& value)
{
   ptr = &value;
   return;
}

int main()
{
   int* a = 0;
   int x = 5;

    // assign a to point to x
    assign(a, x);

    // print out the value 5
   cout << "x: " << *a;

   return 0;
}
```

Listing 4: Assigning a pointer to a value

(4 marks)

[Total Marks 11]