

Assignment 2, Deadline Thurs. 22nd Oct. 2009, 12.30pm (2213). Demonstrations must be completed by Tuesday 20th Oct. at 4.30pm. Worth 30% of CA.

Late assignments without documented extenuating circumstances **will lose 10% of marks per day or part of day late. Those seven days late will receive zero marks.** If you have extenuating circumstances, you should contact me as soon as possible; in particular, be careful about wasting time on completing an assignment only to hand it in well after the deadline; whether you have valid reasons or not I cannot give any marks for work handed in after outline answers have been handed out or discussed in class.

Computing Department Rules. Assignments must typed or neatly handwritten and neatly presented and spelling-checked. Sloppily presented work will be severely penalised or returned for correction and resubmission. Please use the Computing Department cover sheet and signed declaration.

Poor layout of programs will be penalised. You may be questioned about the content of assignments.

Please use only one side of the page. There is no need for ornate front sheets; simply include the heading above (including my name), and your own name. A staple at the top left corner is the preferred binding – I can staple them if you wish. There is no need for plastic or other covering. **For numerical answers, where appropriate, show your working; a numerical answer written down with no explanation of working is likely to get zero marks.**

Software will be available in my public folder.

1. Rewrite BankAccount.java and BankAccountT1.java (with appropriate tests for deposit and withdraw added) in C++.
 - (a) First create BankAccount.h with the function definitions included in the class declaration, i.e. as Cell in page 7-2 of notes. [50 marks]
 - (b) Create BankAccountT1.cpp with `#include BankAccount.h`. Note: when you want to output a BankAccount (e.g. ba1) you will have to use `cout<< ba1.toString();` [50 marks]
 - (c) Demonstration. [50 marks]
2. Rewrite BankAccount.h of Q.1 to BankAccount.cpp (function bodies) and BankAccount.h (class declaration).
 - (a) First create BankAccount.h with the class declaration, i.e. as Cell.h in page 7-4 of notes. [50 marks]
 - (b) Create BankAccount.cpp with the function bodies (has `#include BankAccount.h`), i.e. as Cell.cpp in page 7-4 of the notes. [50 marks]
 - (c) If you do (a) and (b) properly, BankAccountT1.cpp will not need to be changed.
 - (d) Demonstration. [50 marks]

3. Rewrite Bank.java to Bank.cpp (function bodies) and Bank.h (class declaration), and BankTester.java to BankTester.cpp.

Use the BankAccount.h, .cpp from Q. 2.

- (a) Bank.h. Use std::vector for ArrayList.

[50 marks]

- (b) Bank.cpp.

[50 marks]

- (c) BankTester.cpp.

[50 marks]

- (d) Demonstration.

[50 marks]

4. Close relationship between pointers and arrays in C++.

Rewrite arr1.cpp to use pointer notation in blocks A and B.

```
//----- arr1.cpp-----  
// j.g.c. 2009-09-21  
// array example for assignment 2.  
//-----  
#include <iostream>  
using namespace std;  
  
const int n = 10;  
int main(){  
    double a[n];  
  
    // A  
    for(int i = 0; i < n; i++){  
        a[i]= double(i) * 1.1;  
    }  
    cout<< endl;  
  
    // B  
    for(int i = 0; i < n; i++){  
        cout<<" a["<<i<< " ] = "<< a[i]<< ", "  
    }  
    cout<< endl<< endl;  
  
    return 0;  
}
```

- (a) Modified code

[20 marks]

- (b) Demonstration.

[20 marks]

5. Heap allocation in C++.

If we change `arr1.cpp` to have `n` non-const, i.e.

```
int main(){
    int n;
    cout<< "Give n: "<< endl;
    cin >> n;

    double a[n];
```

this will not compile, as `n` must be a const.

Rewrite `arr1.cpp` to use heap allocation (`new`, `delete`).

(a) Modified code

[20 marks]

(b) Demonstration.

[20 marks]

6. `std::vector` versus plain array.

Rewrite `arr1.cpp` to use `std::vector`. Try to change as little code as possible.

(a) Modified code

[20 marks]

(b) Demonstration.

[20 marks]

(c) Write a brief compare and contrast on C++ plain arrays versus `std::vector`. Refer to `std::vector` function `at` versus operator `[]`.

[40 marks]

7. Generating random numbers.

Based on the sample uses of `srand` and `rand` in `dice.cpp` (ch05 programs), write a simple program `dice1.cpp` which will generate a random number in the range 1 ... 45.

(a) Code

[20 marks]

(b) Demonstration.

[20 marks]

8. Generating a Lotto sequence.

Generate a set of six Lotto numbers; forget about bonus numbers. Assume 45 numbers to start with, 44 after first draw, 43 after second, ... This is a bit more difficult than 7; obviously, you need some way of deleting already drawn numbers. One possible way is to use `std::vector` to hold the numbers that are left and to use `erase` when the number is drawn; draw a random number in `1 .. vector.size()`.

(a) Code

[50 marks]

(b) Demonstration.

[50 marks]