1. Namespace? 7
   1. 2- Definition
   2. 3- Syntax
   3. 2- Merge
2. OOP Concept 16
   1. 2- Poly
   2. 2- Inh
   3. 2- Encap, Abs
   4. 2- Abstraction
   5. 2- Inheritance,Encap
   6. 2- Poly
   7. 2- Inh
   8. 2- Inh, Encap
3. Modular? 9
   1. 3 - Def
   2. 2- How and Filenames
   3. 2- Safeghuards, includes
   4. 2- Safeguards Syntax
4. Operator Overload Signature 14
   1. Types Instance 2- Types: Foo, Faa, Fee. Instances: A, B, C
   2. A= B+C 2- Foo operator+(Faa, Fee)   
       Foo Faa::operator+(Fee)
   3. !C 2- bool Fee::operator!()   
       void Fee::operator!();
   4. A = B -=C 2- Foo operator-=(Faa, Fee)   
       Foo Faa::operator-=(Fee)
   5. A = ++C; 2- Foo Fee::operator++();
   6. B = B++; 2- Faa Faa::operator++(int);
   7. C = B = A; 2- Fee Faa::operator=(Foo);
5. Categories of Types: 6
   1. 2- Fundemental, Compound
   2. 4- Primitive types, Made out of other types
6. DMA program: 18

#include <iostream> // essentials 3

using namespace std;

int main() {

int num; // 2 marks

double\* vals;

int lix = 0;

int i;

cout << "Please enter number of doubles values: "; // 1 mark

cin >> num;

vals = new double[num]; // 3 mark

cout << "Please enter the values:" << endl;

for (i = 0; i < num; i++) { // 2 marks

cout << i + 1 << "> ";

cin >> vals[i];

}

for (i = 1; i < num; i++) { // 3 marks

if (vals[i] > vals[lix]) {

lix = i;

}

}

cout << "Values entered: " << endl;

for (i = 0; i < num; i++) { // 1 marks

cout << (i + 1) << ": " << vals[i] <<   
 (i == lix ? " <-- Largest" : "") << endl; // 1 mark

}

delete[] vals; //2 marks

return 0;

}

1. Class design 17 + 7

#ifndef SICT\_SUBJECT\_H\_ // 1 mark

#define SICT\_SUBJECT\_H\_

class Subject { // 1 mark;

char\* m\_title; // 1 marks;

char m\_code[7]; // 2 marks;

int mark;

public:

Subject(); // 1 mark

Subject(const char\* title, const char\* code  
 , int mark = 0); // 4 marks

void print()const; // 2 marks

void read(); // 1 mark

void mark(int value); // 1 mark

int mark()const; // 2 marks;

void operator+=(int mark); // 2 marks

//or

void add(int mark);

const Subject& operator=(const Subject& S); // 3 marks

Subject(const Subject& S); // 2 marks  
 ~Subject(); // 1 mark

};

#endif

1. Bottle 30 + 5

#include <iostream>

using namespace std;

class Bottle {

int m\_capacity; // in CC

int m\_quantity; // in CC, 0 <= m\_value <= m\_capacity

public:

void quantity(int value) { // 2 marks

m\_quantity = value >= 0 && value <= m\_capacity ? value : -1;

}

int quantity()const { // 2 marks

return m\_quantity;

}

int capacity() const { // 2 marks;

return m\_capacity;

}

void empty() { // 1 mark

quantity(0);

}

Bottle(int quantity = 0, int capacity = 500) { // 4 marks

m\_capacity = capacity;

this->quantity(quantity);

}

Bottle& operator+(const Bottle& Ro)const { // 4 marks

return Bottle(m\_quantity + Ro.m\_quantity);

}

Bottle& operator+=(const Bottle& Ro) { // 4 marks]

quantity(quantity() + Ro.quantity());

return \*this;

}

Bottle& operator++() { // 4 marks

quantity(quantity() + 1);

return \*this;

}

Bottle& operator++(int) { // 4 marks

Bottle B = \*this;

quantity(quantity() + 1);

return \*this;

}

operator double()const{ // 3 marks

return double(quantity()) / capacity();

}

};

ostream& operator<<(ostream& ostr, const Bottle& B) { // 5 marks

return ostr << B.quantity() << "CC/" << B.capacity();

}

1. Walk-through 20  
   **2)123  
   2)10  
   2)Yes  
   2)$$  
   2)-----  
   2)----------  
   2)Now what?  
   4)!GOOD MOO!  
   2)321**