Master M1 SMA Computer Programming Class 2012/13.

Exam. 2 hours, open book.

Exercise 1 (1pt)

a) What is a “variable” ?

b) What is the relation between byte and bits ?

c) On 32 bits operating system what is the size in byte of an “int”, a “long int” and a “double” ?

d) Does the range of a “short int” and the range of an “unsigned short int” are the same ? Explain this difference.

Exercise 2 (1pt)

What is the output when the following code fragment is executed ? (see annexe for help)

```
int a,b,c,d,e;
a = 7;
b = 9;
c = 4;
d = 29;
e=  3;
if ( (a+c*d%e)>4 && b/c>1 || (a>b)?(c%a>3):(b%a<2) )
      cout << "out0 : "<< "true" << endl;
else
      cout << "out0 : "<< "false" << endl;
cout << "out1 : "<< ++a << endl;
cout << "out2 : "<< b++ << endl;
cout << "out3 : "<< a++ + ++b << endl;
cout << "out4 : "<< a << endl;
cout << "out5 : "<< b << endl;
cout << "out6 : "<< (c+=e) << endl;
```

Exercise 3 (1pt)

a) What is the output when the following program is executed ?

```
#include <iostream>
using namespace std;
enum weekday
    {monday=1, tuesday, wednesday, thursday, friday, saturday, sunday};
weekday data[]={tuesday,monday,friday,wednesday,sunday};
int main()
{
    for(int i=0; i<7; i++)
    {
      switch (data[i] )
      {
          case monday :
               cout << "m ";
          case tuesday :
               cout << "tu ";
          case wednesday :
               cout << "w ";
          case thursday :
               cout << "th ";
          case friday :
               cout << "f ";
          case saturday :
               cout << "sa ";
          case sunday :
               cout << "su ";
               cout << data[i] << endl;
               break;
          default :
               cout << "prb " << endl ;
               break;
      }
    }
    cout << std::endl;
    return 0;
}
```

b) Is this code is correct ? Explain.

Exercise 4 (4pt)

In this exercise all array are following c++ norm. The declaration must use a static size.

Write a program that ask the user to input two square matrices of integer of the same dimension, term by term (all of them). Matrices can be of any dimension from 1x1 to 10x10.

Internally those two matrices must be stored this way :

- The first one with a unidimensional array.
- The second one with a bi-dimensional array.

In a new step the program must then compute the sum of this two matrices but only for the diagonal terms. The result must be stored in a third array.

In a last step the program print then this last array containing computational result.

For example those 2 “2x2” matrices :

$$\begin{pmatrix} 1 & 7 \\ 4 & 3 \end{pmatrix} \qquad \begin{pmatrix} 3 & 5 \\ 4 & 7 \end{pmatrix}$$

give the following result with this program :

```
  4 10
```

and those 2 “3x3” matrix

$$\begin{pmatrix} 3 & 2 & 0 \\ 4 & 12 & 7 \\ 9 & 2 & 4 \end{pmatrix} \qquad \begin{pmatrix} 6 & 3 & 2 \\ 8 & -2 & 4 \\ 9 & 0 & 9 \end{pmatrix}$$

give :

```
 9 10 13
```

Exercise 5 (2pt)

a) What is the output of the following program ?

```
#include <iostream>
using namespace std;
void sort_bi_0(int a, int b)
{
     if(a>b) { int tmp=b; b=a; a=tmp; }
}
void sort_bi_1(int & a, int & b)
{
     if(a>b) { int tmp=b; b=a; a=tmp; }
}
void sort_bi_2(int * a, int * b)
{
     if(*a>*b) { int tmp=*b; *b=*a; *a=tmp; }
}
void sort_bi_3(int ** a, int ** b)
{
     if(**a>**b) { int *tmp=*b; *b=*a; *a=tmp; }
}
int main(){
   int c =4,d=2;
   int *pc=&c;
   int *pd=&d;
   cout << c << " "<< d << " " <<*pc<< " " <<*pd<< endl;
   sort_bi_0( c,d);
   cout << c << " "<< d << " " <<*pc<< " " <<*pd<< endl;
   c =4;d=2;
   sort_bi_1( c,d);
   cout << c << " "<< d << " " <<*pc<< " " <<*pd<< endl;
   c =4;d=2;
   sort_bi_2( &c,&d);
   cout << c << " "<< d << " " <<*pc<< " " <<*pd<< endl;
   c =4;d=2;
   sort_bi_3( &pc,&pd);
   cout << c << " "<< d << " " <<*pc<< " " <<*pd<< endl;
   return 0;
}
```

b) Explain

Exercise 6 (1pt)

What is the output of the following program ?

```
#include <iostream>
using namespace std;
int main()
{
     int a[] = {1,3,5,7,11};
     int *p=a;
     int *q=&a[2];
     cout <<  *(p++) << " " << ++*q << endl;
     cout <<  *++p << " " << *(q++) << endl;
     cout << p[1] << " " << q[1] << endl;
     cout << q-p <<endl;
     cout <<a[0]<<" "<<a[1]<<" "<<a[2]<<" "<<a[3]<<" "<<a[4]<<endl;
     return 0;
}
```

Exercise 7 (2pt)

Bellow is given the header geom3D.h

```
1       class geom3D
2       {
3         public :
4          virtual bool isOn(const double &x, const double &y, const double &z) const = 0;
5          virtual void isOfType(void) const =0;
6       };
7       class plane : public geom3D
8       {
9         public :
10          plane(const double &a_,const double &b_, const double &c_, const double &d_);
11          bool isOn(const double &x, const double &y, const double &z) const;
12          void isOfType(void) const;
13        private :
14          double a,b,c,d;
15      };
16      class point : public geom3D
17      {
18        public :
19          point(const double &x_,const double &y_, const double &z_);
20          bool isOn(const double &x, const double &y, const double &z) const;
21          void isOfType(void) const;
22        private :
23          double px,py,pz;
24      };
25      class line : public geom3D
26      {
27        public :
28          line(const double &x_,const double &y_, const double &z_,const double
        &dx_,const double &dy_, const double &dz_);
29          bool isOn(const double &x, const double &y, const double &z) const;
30          void isOfType(void) const;
31        private :
32          double dp[6];
33      };
```

This is describing a simple geometric class hierarchy. Base on those declarations answer to the following questions.

a) The following fragment of code is not compiling ? Explain. What have to be changed in geom3D.h to make it compile.

```
plane pla(2.,1.,2.,-2.);
cout<<pla.d<<end;
```

b) Line 21 what is the meaning of the const key word.

c) From a vocabulary point of view fill the #blank# :

“geom3D” is the #blank# class of “point” class.

”line” is a #blank# class of “geom3D”

“line” class #blank# all members from class “geom3D”. Saying all public/protected member of class “geom3D” are accessible in class “line”.

d) Does the following fragment of code is possible ? Explain.

```
geom3D b;
```

Exercise 8 (3pt)

This exercise use header geom3D.h of previous exercise.

It is used in the following program :

```
#include <iostream>
#include "geom3D.h"
using namespace std;
int main ()
{
  double x_1=0.;
  double y_1=0.;
  double z_1=0.;
  double x_2=0.;
  double y_2=0.;
  double z_2=1.;
  double x_3=1.;
  double y_3=0.;
  double z_3=0.;
  double x_4=1.;
  double y_4=1.;
  double z_4=1.;
  plane pla(2.,1.,2.,-2.);
  point p1(x_1,y_1,z_1);
  line l1(x_1,y_1,z_1,1.,0.,1.);
  line l2(1.,1.,1.,1.,1.,0.);
  geom3D * entities[4]={&pla,&l1,&l2,&p1};
  for(int i=0; i<4; ++i)
  {
       cout<<"=================================================="<<endl;
       entities[i]->isOfType();

       cout<<"Point at location ("<< x_1 <<","<<y_1<<","<<z_1<<") "<<endl;
       entities[i]->isOn(x_1,y_1,z_1)?cout<<"is on ":cout<<"is not on ";
       cout <<"entity"<<endl;

       cout<<"Point at location ("<< x_2 <<","<<y_2<<","<<z_2<<") "<<endl;
       entities[i]->isOn(x_2,y_2,z_2)?cout<<"is on ":cout<<"is not on ";
       cout <<"entity"<<endl;

       cout<<"Point at location ("<< x_3 <<","<<y_3<<","<<z_3<<") "<<endl;
       entities[i]->isOn(x_3,y_3,z_3)?cout<<"is on ":cout<<"is not on ";
       cout <<"entity"<<endl;

       cout<<"Point at location ("<< x_4 <<","<<y_4<<","<<z_4<<") "<<endl;
       entities[i]->isOn(x_4,y_4,z_4)?cout<<"is on ":cout<<"is not on ";
       cout <<"entity"<<endl;
  }

  return 0;
}
```

Write an implementation of the class (in a geom3D.cc) to be able to compile the above program an obtain the following output on the terminal :

```
==================================================
Entity is of type plane
Point at location (0,0,0)
is not on entity
Point at location (0,0,1)
is on entity
Point at location (1,0,0)
is on entity
Point at location (1,1,1)
is not on entity
==================================================
Entity is of type line
Point at location (0,0,0)
is on entity
Point at location (0,0,1)
is not on entity
Point at location (1,0,0)
is not on entity
Point at location (1,1,1)
is not on entity
==================================================
Entity is of type line
Point at location (0,0,0)
is not on entity
Point at location (0,0,1)
is on entity
Point at location (1,0,0)
is not on entity
Point at location (1,1,1)
is on entity
==================================================
Entity is of type point
Point at location (0,0,0)
is on entity
Point at location (0,0,1)
is not on entity
Point at location (1,0,0)
is not on entity
Point at location (1,1,1)
is not on entity
```

To do so you have to implement 9 members functions.

The member “isOn” is supposed to answer true if the point, which coordinates are given in argument, is geometrically on the object.

The following analytical geometry procedure must be followed :

- Every computation are done in the 3D Cartesian coordinate of axes x, y and z.
- The class plane is storing the coefficients of the equation a*x+b*y+c*z+d=0 to define an instance. A point is on this plane if its coordinate verify this equation.
- The class point is storing the coordinate px, py and pz to define an instance. A point is on a

point if all their coordinates are identical.

- The class line is storing coordinate of a point on this line (x_,y_,z_) and a vector (dx_,dy_,dz_) collinear to it, to define an instance. All those informations are stored in the data class member dp. A point of coordinate (x,y,z) is on this line if it exist a real number “t” which verify the following set of equations :
  - x=x_+t*dx
  - y=y_+t*dy
  - z=z_+t*dz

Exercise 9 (1pt)

This exercise use header geom3D.h of previous exercise.

Change this header to add the appropriate declaration of a function “interPlaneLine” to compute the intersection of a plane and a line given as 2 parameters :

- The reference to an instance of the class “plane”.
- The reference to an instance of the class “line”.

This function return a boolean which is true if there is an intersection.

No implementation of this function is expected in this exercise but to answer the question you have to consider that the implementation is supposed to use member data of the objects given in argument.

In geom3D access status to member of class must not be changed and you must not add any new methods.

You will certainly need to use an incomplete class declaration. It is a class declaration that does not define any class members but help use reference or pointer of this class prior to it's complete declarations.

Here is an example of an incomplete class declaration usage :

```
class A;
class B
{
 public:
  A *pa;
  void foo(const A& k);
};

class A  {
 public:
    ......
};
```

Exercise 10 (2 pts)

Here is a simple function template :

```
template <typename T>
void mysort (T & a, T & b)
{
    if (a>b)
    {
        T tmp(a);
        a=b;
        b=tmp;
     }
};
```

a) What are the minimum set of operator/member function a class need, to be able to be use (as template parameter) by the function mysort ?

b) Write a main program that use the function mysort to sort two double.

Exercise 11 (1pts)

What is the best STL container to use in those circumstances :

a) I want to store integer. I want a fast access and I want to be sure that no duplication occurs.

b) I want to store a huge set of object indexed by a unique integer. I want also to have an extremely fast access to these objects via index. I don't care how long it will take to store information in this container and i will never iterate on it.

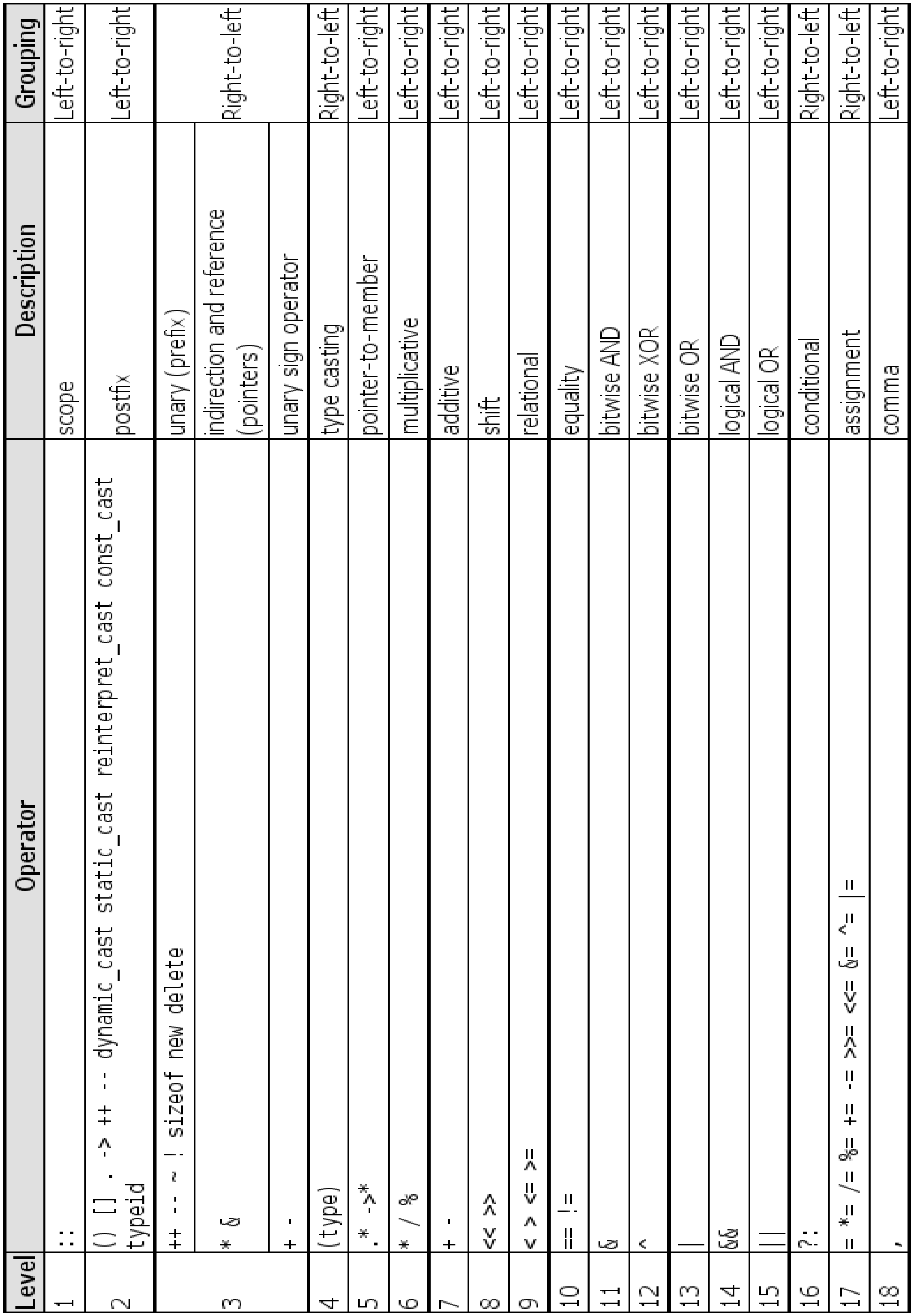
c) I want to add and remove quickly informations from any place in my container.

d) After filling container element by element, i want to iterate many times on these informations in a random manner.

Exercise 12 (1pts)

What are “**doxygen**”, “**make**” and “**gdb**”. Explain at least their purpose and the general way to use them.

Annexe : **Precedence**



Level	Operator	Description	Grouping
1	::	scope	Left-to-right
2	() [] . -> ++ -- dynamic_cast static_cast reinterpret_cast const_cast typeid	postfix	Left-to-right
3	++ -- ~ ! sizeof new delete	unary (prefix)	Right-to-left
	* &	indirection and reference (pointers)	
	+ -	unary sign operator	
4	(type)	type casting	Right-to-left
5	.* ->*	pointer-to-member	Left-to-right
6	* / %	multiplicative	Left-to-right
7	+ -	additive	Left-to-right
8	<< >>	shift	Left-to-right
9	< > <= >=	relational	Left-to-right
10	== !=	equality	Left-to-right
11	&	bitwise AND	Left-to-right
12	^	bitwise XOR	Left-to-right
13	|	bitwise OR	Left-to-right
14	&&	logical AND	Left-to-right
15	||	logical OR	Left-to-right
16	?:	conditional	Right-to-left
17	= *= /= %= += -= >>= <<= &= ^= |=	assignment	Right-to-left
18	,	comma	Left-to-right