Task1. What is the output for the following program?

```
#include "iostream.h"

int dat=1;
class KlasaB;

class B {
public:
	virtual void f(int d=5)=0;
};

class A {
	unsigned i;
public:
	int pod;

	A& operator++() { cout << "A " << pod-- << dat++ << endl; A* b = new A; return *b;}
	A operator++(int) { cout << "B " << pod++ << dat-- << endl; A* b = new A; return *b;}
	A& operator--() { cout << "C " << dat++ << ++pod << endl; A* b = new A; return *b;}
	A operator--(int) { cout << "D " << dat--<< --pod << endl; A* b = new A; return *b;}
	void operator+(A& b) { pod++;}

	friend void B::f(int);
	 A() : i(0), pod(dat) {pod++; cout << "CA" << endl;}
	~A(){pod++; cout << "DC" << endl;}
};

const KlasaB& returnKlasaA(const KlasaB& p){return p;}


class KlasaB : public A, public B {
 public:
	 void f(int d=2){ A c; c++; pod*=d--;}
	 KlasaB(){cout << "CB" << endl;}
	 ~KlasaB(){cout << "DB" << endl;}
};

KlasaB returnKlasaB(KlasaB s){ return s;}

class KlasaC : public KlasaB {
public:
	KlasaC(){cout << "CC" << endl;}
	~KlasaC(){cout << "DC" << endl;}
};

KlasaB returnKlasaC(KlasaB s)
{
	return s;
}

void main(int argc, char* argv[])
{
	KlasaC x;
	A v,w;
	B *c = new KlasaB;
	--++v--+++w--;
	returnKlasaC(returnKlasaB(returnKlasaA(x)));
}
```

Task 2. What is the output for the following program?

```
#include <iostream.h>

class Base
{
    int x;
    virtual int f( int );
public:
    Base( int a ) : x(a) {}
    double f( double );
};

int Base::f( int a )
{
    return a+x;
}

#define KUB(a) a*a*a

double Base::f( double a )
{
    cout << "AA";
    return KUB(a*x-3) + f(5);
}

class Derived : public Base
{
    int x;
    int f( int );
public:
    Derived ( int a, int b ) : Base(a), x(b) {cout << "BB";}
    double f( double );
};

int Derived::f( int a )
{
    cout << "CAB";
    return KUB(a-x);
}

double Derived::f( double a )
{
    cout << "BAC" <<  f(2);
    return a/x+f(7);
}

void main()
{
    Derived d(3,5);
    cout << d.f( 3.4 ) << endl;
    Base* pb = new Derived(2,4);
    cout << pb->f( 4.5 ) << endl;
    Base b(2);
    cout << b.f( 2.3 ) << endl;
}
```

Task 3. What is the output for the following program?

```
#include <iostream>
#include <complex>

using namespace std;

class Base{
public:
  virtual void f( int );
  virtual void f( double );
  virtual void g( int i = 3 );
};

void Base::g( int i ){  cout << "XXX" << i << endl;}

void Base::f( int ){ cout << "ABC" << endl;}

void Base::f( double ){ cout << "CACA" << endl;}

class Derived: public Base{

public:
  void g( int i = 2 );
  void f( complex<double> );
};

void Derived::f( complex<double> ){ cout << "CBA" << endl;}

void Derived::g( int i ){
       cout << "BABC**" << i << endl;
       Base b = *(new Derived());
       try {
              if (i==2) throw Derived();
              if (i==3) throw 7.6;
              if (i==4) throw 2.4f;
              if (i==5) throw b;
       }
       catch (Base) {cout << "ASS";}
       catch (double) {cout << "ATT";}
       catch (Derived) {cout << "AMM";}
       catch (float) {cout << "AVV";}
}


void main(){
  Base    b;  Derived d;  Base*   pb = new Derived;

  d.f(5.0);
  pb->f(6.0);
  b.g();
  b.g(5);
  d.g();
  b.f(4.0);
  pb->g();
  pb->g(4);
  delete pb;
}
```

Task 4. What is the output for the following program?

```
#include "iostream.h"

static int b;

class X {
	int *pi;

public:

	int osa, buba, mis;

	X() {b=0; osa=3; mis = 5*b++; buba=mis++;};
	X(const X &x) : pi(new int(*x.pi)){b*=2; osa=-2; mis = -4*b++; buba=mis--;}
	X(int l) : pi(new int(l)), osa(l), mis(l++), buba(l++) { b++;}
	X& operator= (const X&);
	~X() {cout << "b:"<< b << "osa:"<< osa << "buba:"<<buba << "mis:"<<mis << endl;delete
pi;}
};

X& X::operator= (const X& x) {
	if (this != &x) {
		delete pi;
		pi = new int(*x.pi);
	}
	return *this;
}

X funF(X x)
{
	X Xnew(5);
	x=Xnew;
	Xnew.osa++; Xnew.mis+= ++b; Xnew.buba=b;
	return x;
}

void g() {

	X xa=3, xb=1;
	X xc = xa;
	xa = funF(xb);
	xc = xa;
	X a(2);
	a.buba++;
	b+=a.buba;
}

void main(int argc, char* argv[])
{
	X d();
	g();
}
```

Task 5. What is the output for the following program?

```
#include <iostream.h>

unsigned long address = 0;
int dword = 8;
int dd   = 7;
int word  = 4;
int byte  = 1;

class A
{
public:
unsigned long operator+() {cout << "opA+()" << endl; return address+word;}
unsigned long operator+(unsigned long) {cout << "opA+(ul)" << endl; return address+word;}
unsigned long operator+(A) {cout << "op+(A)" << endl; return address+dd;}
unsigned long operator-() {cout << "opA-()" << endl; return address-dword;}
unsigned long operator-(unsigned long) {cout << "opA-(ul)" << endl; return address-word;}
      unsigned long operator-(A) {cout << "op-(A)" << endl; return address-dd;}
};

class B
{
public:
unsigned long operator-() {cout << "opB-()" << endl; return address-dword;}
unsigned long operator-(unsigned long) {cout << "opB-(ul)" << endl; return address-word;}
unsigned long operator-(B) {cout << "op-(B)" << endl; return address-dd;}
unsigned long operator+() {cout << "opB+()" << endl; return address+word;}
unsigned long operator+(unsigned long) {cout << "opB+(ul)" << endl; return address+word;}
unsigned long operator+(B) {cout << "op+(B)" << endl; return address+dd;}
};

unsigned long operator+(A a) {cout << "op+(A)ext" << endl; return address+dd+6;}
unsigned long operator+(unsigned long g, A a) {cout << "op+(ul,A)ext" << endl; return g+6;}
unsigned long operator-(A a) {cout << "op-(A)ext" << endl; return address-dd-7;}
unsigned long operator-(unsigned long g, A a) {cout << "op-(g,A)ext" << endl; return g-7;}

void main()
{
      A a, d, e, m, n;
      B g, h;
      unsigned long b, c, f;
      address = 30;
      b = 3U; c = 5U; f = 1U;
      B* i = (B*)new A();
      cout << b-e-m+b+(*i+g)+n-(a+c)+(h-(d-f)) << endl;
}
```

Task 6. What is the output for the following program?

```
#include "stddef.h"
#include "iostream.h"

template <class Xtype, int SD>
class Node
{
public:
    int axis;
    Xtype x[SD];
    unsigned int id ;
    Node* Parent;
    Node* Left;
    Node* Right;

    Node() {};
    virtual ~Node() {};
    Node(Xtype* x0, int split_axis);
    Node*    Insert(Xtype* x);
    Node*    FindParent(Xtype* x0);
    int      Traverse(Xtype* xS, int level);
};

template <class Xtype, int SD> Node<Xtype, SD>::Node(Xtype* x0, int split_axis)
{
    for (int i=0; i<SD; i++)
      x[i] = x0[i];
    axis = split_axis;
    Parent = NULL;
    Left = NULL;
    Right = NULL;
}

template <class Xtype, int SD> Node<Xtype, SD>* Node<Xtype, SD>::FindParent(Xtype* x0)
{
    Node* parent;
    Node* next = this;
    int split;
    while(next) {
        split = next->axis;
        parent = next;
        if(x0[split] > next->x[split])
            next = next->Right;
        else
            next = next->Left;
    }
    return parent;
}

template <class Xtype, int SD> Node<Xtype, SD>* Node<Xtype, SD>::Insert(Xtype* p)
{
    Node* parent = FindParent(p);
    Node* newNode = new Node(p, parent->axis+1 < SD ? parent->axis+1:0);
    newNode->Parent = parent;
    if(p[parent->axis] > parent->x[parent->axis])
        parent->Right = newNode;
    else
        parent->Left = newNode;
    return newNode ;
}
```

```
template <class Xtype, int SD>
int Node<Xtype, SD>::Traverse(Xtype* xS, int level)
{
	int ret = 0;
	if (axis%2) {
		if (x[axis] > xS[axis]) {
			if (x[(axis+1)%SD] < xS[(axis+1)%SD])
				ret++;
			if (Left != NULL)
				ret += Left->Traverse(xS, ++level);
		}
		if (Right != NULL)
			ret += Right->Traverse(xS, ++level);
	} else {
		if (x[axis] < xS[axis]) {
			if (x[(axis+1)%SD] > xS[(axis+1)%SD])
				ret++;
			if (Right != NULL)
				ret += Right->Traverse(xS, ++level);
		}
		if (Left != NULL)
			ret += Left->Traverse(xS, ++level);
	}
	cout << "A_" << x[axis] << " B_" << x[(axis+1)%SD] << endl;

	return ret;
}

int main(int argc, char* argv[])
{
	int x[2];
	x[0] = 5;	x[1] = 8;
	Node<int, 2> tree(x, 0);
	x[0] = 3;	x[1] = 5;
	tree.Insert(x);
	x[0] = 10;	x[1] = 7;
	tree.Insert(x);
	x[0] = 15;	x[1] = 4;
	tree.Insert(x);
	x[0] = 1;	x[1] = 8;
	tree.Insert(x);
	x[0] = 6;	x[1] = 9;
	tree.Insert(x);
	x[0] = 2;	x[1] = 3;
	tree.Insert(x);
	x[0] = 7;	x[1] = 4;
	cout << tree.Traverse(x, 0) << endl;
	x[0] = 4;	x[1] = 4;
	cout << tree.Traverse(x, 0) << endl;
	return 0;
}
```

Answers: Object oriented programming – Budva 2009

Task1 (16):	Task6 (20):
Task2 (16):	Task5 (16):
Task5 (16):	Task4 (16):

Task1	Task2	Task3	Task4	Task5	Task6	Sum()