<pre>x =a; y =b; sumX +=9 if(y&gt;temp    yInd =z } else temp } static void ge</pre>	){ ; =y;	//	for displa	ay data mel	mber z				
<pre>void getAll()     cout&lt;&lt;"x         "y = "&lt;         "z = "&lt;- }  void getConcout&lt;&lt;"x         &lt;&lt;"y =         &lt;&lt;"z =         &lt;&lt;"z =         &lt;&lt;"z =         &lt;&lt;"z =         &lt;&lt;"color</pre>	= "< <x<<end <y<<endl<< <z<<endl;< th=""><th>st{ dl</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></z<<endl;<></y<<endl<< </x<<end 	st{ dl							
<pre>} static void S    cout&lt;&lt;"th } static void Y</pre>	umOfx() { e sum of all	x = "< <si< th=""><th></th><th></th><th><endl;< th=""><th></th><th></th><th></th><th></th></endl;<></th></si<>			<endl;< th=""><th></th><th></th><th></th><th></th></endl;<>				
<pre>}; int Test::z = 0; int Test::sumX int Test::yInd= int Test::temp = int main(){     Test a;     Test b(4,5);     Test c(b);     Test d(8, 7);     Test e;     e.Initilize(7,     e.getConstD      Test::SumOf     Test::YmaxI      return 0; }</pre>	9); ata(); x();								
OUTPUT:  x = 7 y = 9 z = 5 the sum of all x Y's maximum	alue is class	1	MODUI						
Table: inheritance  Class  Son  public  protected  private  public  protected  private  public  protected  private  public  protected  private  public	Grandsor public public public protected protected private private		In son class money  NO  NO  NO  NO  NO  NO  NO  NO  NO  N		land YES		In Grandso Money NO	on class Gold YES YES NO YES NO YES NO YES NO YES NO YES NO	Land YES YES NO YES NO YES YES NO YES NO YES NO
Topic: Types of In  1. Single Inher  class A{ private:     int x; protected:	nheritance	// x is n void ş	:public A{ ot accessibl getz(){ cout	le t<<"z is " <<	z< <endl;< td=""><td>ii F b</td><td>nt main(){ 3 b; o.getz(); o.gety();</td><td></td><td></td></endl;<>	ii F b	nt main(){ 3 b; o.getz(); o.gety();		
int y; public: int z; }		}	class A private: int x; protected int y; public: int z;  Class B only y and z access	A d:	< <y<<endl< td=""><td>;   r }</td><td>eturn 0;</td><td></td><td></td></y<<endl<>	;   r }	eturn 0;		
class A{ private:     int x; protected:     int y; public:     int z; }	I Inheritance:	class B:pul	blic A{	E  //   <	lass C:public 3{ / x is not acce void getz(){ < <z<<endl; c<y<<endl;<="" gety(){="" td="" void=""><td>essible { cout&lt;&lt;"</td><td>C</td><td>t main(){ c; getz(); gety(); turn 0;</td><td></td></z<<endl;>	essible { cout<<"	C	t main(){ c; getz(); gety(); turn 0;	
			private: int x; protected: int y; public: int z;	Class B and z are eccessible  Class c are accessible	e				
Class A{ private: int x; protected: int y; public: int z; }		class B{ private: int p; protected: int q; public: int r; }		P	class C:public Public B{ void getz(){ c <z<<endl; <<q<<endl;="" <<r<<endl;="" <<r<<endl;<="" <q<<endl;="" <y<<endl;="" c="" getq(){="" getr(){="" gety(){="" roid="" td="" void=""><td>cout&lt;&lt;"y out&lt;&lt;"q i</td><td>is " C. c. re is " }</td><td>t main(){ c; getz(); gety(); turn 0</td><td></td></z<<endl;>	cout<<"y out<<"q i	is " C. c. re is " }	t main(){ c; getz(); gety(); turn 0	
i I I	Class A private: nt x; protected: nt y; public: nt z;			Class c		Class private: int p; protected int q; public: int r;	В		
4. Heirarchical			y,z,p,c	od p are not sible  q are accessil	class C:publi	cessible	B C	nt main(){ b; c; getz():	
<pre>private: int x;</pre>		void getz( < <z<<end< td=""><td>){</td><td></td><td>void getz(){ c &lt;<z<<endl; void gety(){ c &lt;<y<<endl;< td=""><td></td><td>is " b. b. c.; is " c.;</td><td>getz(); gety(); getz(); gety(); turn 0;</td><td></td></y<<endl;<></z<<endl; </td></z<<end<>	){		void getz(){ c < <z<<endl; void gety(){ c &lt;<y<<endl;< td=""><td></td><td>is " b. b. c.; is " c.;</td><td>getz(); gety(); getz(); gety(); turn 0;</td><td></td></y<<endl;<></z<<endl; 		is " b. b. c.; is " c.;	getz(); gety(); getz(); gety(); turn 0;	
private: int x; protected: int y; public: int z; }		< <y<<end< td=""><td>Class A</td><td>}</td><td></td><td></td><td></td><td></td><td></td></y<<end<>	Class A	}					
int x; protected: int y; public: int z;	Class Only y and accessible	<pre></pre>	Class A	}		Class C			
int x; protected: int y; public: int z; }  5. Hybrid Inher class A{ private: int x; protected: int y; public: int z;	only y and accessible  itance:  class in public of the state of the st	<pre> <th>Class A ivate: i x; otected: i y; blic: i z;</th><th>lass C: virtu</th><th>acces ual public I tual , d guous</th><th>class D:pB, public // x is novoid getz // void gety</th><th>oublic c C { t accessible (){ cout&lt;&lt;"z</th><th>z is d.getz return</th><th>r(); ();</th></pre>	Class A ivate: i x; otected: i y; blic: i z;	lass C: virtu	acces ual public I tual , d guous	class D:pB, public // x is novoid getz // void gety	oublic c C { t accessible (){ cout<<"z	z is d.getz return	r(); ();
int x; protected: int y; public: int z; }  5. Hybrid Inher class A{ private: int x; protected: int y; public:	only y and accessible  itance:  class i public { // with will b	exyxend  pri int pro i	Class A ivate: i x; otected: i y; blic: i z;	lass C: virtument virtumen	acces ual public I tual , d guous	class D:pB, public // x is novoid getz // void gety	oublic c C { t accessible (){ cout<<"? ndl; r(){ cout<<"?	D d; d.gety z is d.getz return	r(); ();
int x; protected: int y; public: int z; }  5. Hybrid Inher  class A{ private: int x; protected: int y; public: int z; };	only y and accessible  itance:  class i public { // with will b	B: virtual a continuation of the continuation	Class A ivate: i x; otected: i y; blic: i z; class A ivate: i x; otected: i y; blic: i z;	lass C: virtument virtumen	acces  ual public  itual , d guous	class D:pB, public // x is no void getz // < <z<<er> void gety // &lt; class D:pB, public // x is no void gety // &lt;<z<<er> void gety // &lt;&lt;</z<<er></z<<er>	oublic c C { t accessible (){ cout<<"? ndl; r(){ cout<<"?	D d; d.gety d.getz return }	r(); ();
int x; protected: int y; public: int z; }  5. Hybrid Inher class A{ private: int x; protected: int y; public: int z; };  Or	itance:  class public { // with will b };  Class B ally y and z are a	eccessible  B: virtual put into put int	Class A ivate: i x; otected: i y; blic: i z; class A {	Class A e: ;; cted: y; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;	accessible of B and C at a same parent	class D:pB, public (/ x is not void getz ' < <z <<e=""> class C:pB, public (/ x is not void getz ' &lt;<z <<e=""> class C:pB, public (/ x is not void getz ' &lt;<z <<e=""> class C:pB, public (/ x is not void getz ' &lt;<z <<e=""> class C:pB, public (/ x is not void getz ' &lt;&lt;</z></z></z></z>	oublic c C { t accessible (){ cout<<"? ndl; cy< <endl;< td=""><td>D d; d.gety d.getz return }</td><td>r(); ();</td></endl;<>	D d; d.gety d.getz return }	r(); ();
Topic: [Con  1.Single Inh  class A { private: int x; protected: int y; public: int z; };  Or  Or	class B aly y and z are a	B: virtual printing put into p	Class A ivate: ix; otected: ix; otected: ix; blic: ix z; blic: int x protec int y public int z  class B { private int z protec int y public int z public:	Class C: virtue without virgill be ambiguited;  ass A e: cted: y; ;; ;; in Inherited from the control of the co	accessible tual, d guous cessible of B and C an same parent.	class D:pB, public // x is not void getz // < <z<<ed>class D:pB, public // x is not void getz // &lt;<z<<ed>class D:pB, public // x is not void getz // &lt;<z<<ed>class D:pB, public // x is not void getz // &lt;<z<<ed>class D:pB, public // x is not void getz // &lt;</z<<ed></z<<ed></z<<ed></z<<ed>	oublic c C { t accessible (){ cout<<"? ndl; cy< <endl;< td=""><td>D d; d.gety d.getz return } sible</td><td>r(); ();</td></endl;<>	D d; d.gety d.getz return } sible	r(); ();
5. Hybrid Inher class A { private: int x; protected: int y; public: int z; };  Topic: [Con  1. Single Inh  class A { private:     int ax;  public:     A(int a)     {         ax = a;         cout <<     endl;     }     int getax()     {         return a     }     ~A()     {	Class B ally y and z are a structor are are are are are are are are are ar	B: virtual print put int print put int print put int p	Class A ivate:  It is	Class D without vir without vir rill be ambig street: y; the control c	accessible tual, d guous as a same parent same parent same parent same parent same parent same same parent same same same same same same same same	class D:pB, public // x is norvoid getz // self // x is norvoid gety // self /	ant main( {     B b(10 b.sum(     return)	D d; d.gety d.getz return } sible	r(); ();
int x; protected: int y; public: int z; };  Class A{ private: int x; protected: int y; public: int z; };  Class A { private: int ax;  public: A(int a) {     ax = a;     cout <<     endl;     }     int getax()     {         return a     }     ~A()     {         cout <<     endl;     } };  OUTPUT: constructor A constructor B sum = 20 Destructor B sum = 20 Destructor B	class public { // with will b };  Class B ally y and z are a	B: virtual print put int print put int print put int p	Class A ivate:  ivate: ix; otected: ix; blic: ix; protection int y public: int y public int z  Class B { private int y public int y public: int by co set endl } void { co set endl } end } end set endl end set endl end	Class D without vir ill be ambig is  ass A e: ; cted: y; ; ty; ; ty; ty; ty; ty; ty; ty; ty;	accessible tual, d guous as a same parent same parent same parent same parent same parent same same parent same same same same same same same same	class D:pB, public // x is not void getz // self self self self self self self self	ant main( {     B b(10 b.sum(     return)	D d; d.gety d.getz return } sible	r(); ();
Int x; protected: int y; public: int z; }  S. Hybrid Inher  class A {   private:   int x;   public:   int z; };  Or  1.Single Inh  class A {   private:   int ax;  public:   A (int a) {     ax = a;     cout <<     endl;     }   int getax()     {       return a     }     ~A()     {       cout <<     endl;     } };  OUTPUT: constructor A constructor B constructor B constructor B constructor B constructor B constructor A  Constructor A  Sequence of the constructor	only y and accessible  class public for an eritance:  Class B ally y and z are a eritance:  are constructor  itance:  Class B ally y and z are a eritance:  constructor  itance:  constructor  itance:	B: virtual B: virtual Anout virtual e ambigue accessible  care  care  for A''  ca	Class A ivate:  ivate:	Class C: virtue without vir without vir ill be ambig ass A e: cted: y; ty	accessible of B and C at a same parent	class D:p B, public // x is not void getz void gety self; Class Class D:p B, public // x is not void gety self; r B"  Self B'  Se	ant main( {     t accessible (){ cout<";     total cout in the cou	D d; d.gety d.getz return }  Sible	estructor A.
int x; protected: int y; public: int z; }  S. Hybrid Inher class A{ private: int x; protected: int y; public: int z; };  Or  1.Single Inh  class A { private: int ax;  public:     A(int a)     {         ax = a;         cout <<         endl;         }         int getax()         {             cout <<         endl;         }         *A()         {             cout <<         endl;         }         *Asymmetric constructor A constructor B constructor B constructor B constructor B constructor A  Sequence of the constructor A	itance:    class   public   will b     will b	B: virtual A print and Dest and Dest and Dest are ambigue are are be are constructed by are constructed are constructed are constructed by are constructed are constructed by are constructed are constructed by are constructed by are constructed are constructed by are constructed	Class A ivate:  Class A ivate:  ivate:  ix six ix i	Class D. without virill be ambiguated from the constructor of the cons	itance]  cessible of B and C at a same parent  and	class D:p B, public y x is not void getz void gety side side  Cla only y and  S B(c) { Cla only y and  A class D:p B, public y x is not void gety side side  Cla only y and  S Cla only y and  Cla only y and  S Cla only y and  Cla only y and  S Cla only y and  Cla only y and  S Cla only y and  Cla only	ant main( {	D d; d.gety d.getz return  y is  or B() → D  or main()  C c(5); c.sum()	estructor A.
int x; protected: int y; public: int y; public: int z; }  S. Hybrid Inher class A { private: int x; public: int z; }; }  Class A { private: int x; public: int z; }; }  OUTPUT: constructor A constructor A constructor B construc	itance:  class B ly y and z are a  eritance:  execution:  "Constructor  at eritance:  at eritance:  at eritance:  at execution:  at execution	B: virtual  A pout virtual  B: A  B: virtual  A nout virtual  Constructor  Construc	Class A ivate:  Class A ivate:  ix protect int ix protect int ix protect int ix ix protect int ix protect int ix	Class C: virtue without vir without vir will be ambig stated: y; ass A e: y; ass A e: y; ass A e: y; the by: A (b)  A (c) y and z are ac al child class anherited from the constructor by the constructor b	itance]  cessible of B and C acts are parent  cestructor  and a same parent  cestructor  c	class D:p B, public y x is not void getz void gety side side  Class Only y and  Cla only y and	are  oublic C { t accessible (){ cout<"; ty <endl; a="" access="" are="" ass="" c="" country="" td="" the="" to="" ty="" ty<="" ty<endl;="" z="" {=""><td>D d; d.gety d.getz return  y is  or B() → D  or main()  C c(5); c.sum()</td><td>estructor A.</td></endl;>	D d; d.gety d.getz return  y is  or B() → D  or main()  C c(5); c.sum()	estructor A.
int x; protected: int y; protected: int y); public: int z; }  5. Hybrid Inherical Class A { private: int z; protected: i	class B public will b structor at eritance:  "constructor at eritance:  "x;  "destructor  ax;  "destructor  ax;  "destructor	B: virtual print p	Class A ivate:  Class A ivate:  int x protected:  int x protected:  int x public:  B(int bx col int z int x public:  int y int z int x int x public:  int y int x	ass C: virtue without vir without vir without vir rill be ambig serviced: y; state ();	itance]  Coessible of B and C as a same parent as a same	sum() =  Sum	ant main( {	D d; d.gety d.getz return  y is  Sible  Or B() → D  Or B() C c(5); C.sum() return Or  Or B()	estructor A.
Int x; protected: int y; protected: int z; }  S. Hybrid Inher class A { private: int y; public: int z; }; protected: int y; public: int z; }; public: int z; }; public: int getax() { ax = a; cout << endl; } int getax() { cout << endl; } ;  Y. A() { cout << endl; } ;  Y. Sequence of the class A { private: int ax; public: A (int a) { ax = a; cout < cut constructor B constructor B constructor A constructor A constructor B constructor B constructor C sum = 15 Destructor C sum = 15 Destructor A constructor B constructor B constructor B constructor B constructor C sum = 15 Destructor A constructor B cons	itance:    Class B   public	Constructed and Design and Design and Design and Secretary	Class A  ivate:  ix;  Class A  ivate:  ix;  otected:  ix;  otected	Class C: virtue without virill be ambigued ass A e: class C: without virill be ambigued ass A e: class C: class D and Inheria class A e: class C: c	cessible of Band Can same parent of	class D:p B, public X is not Y and Z Sible  Class D:p B, public Y x is not Y void getz Y v	are  Dublic C {     t accessible (){ cout<''; ndl; (){ cout<'';     ty <endl; of="" second="" td="" the="" the<=""><td>D d; d.gety d.gety return y is  or B() → D  or main() C c(5); c.sum() return 0  C c(5); return 0;</td><td>O; (); (); (); (); (); (); (); (); (); ()</td></endl;>	D d; d.gety d.gety return y is  or B() → D  or main() C c(5); c.sum() return 0  C c(5); return 0;	O; (); (); (); (); (); (); (); (); (); ()
Int x; protey:	itance:   Class   Clas	Constructs  B: virtual B: virtual Cass B A care  Class B A control B: A B: virtual Constructs  Class B A control Constructs  C	Class A (and and and and and and and and and and	A Class C: virtue without virial be ambiguing ass A e: cted: y; ass A e: tted: y; structor ass A e: tt	cal public of a pu	class D:p B, public X is not Y ovid getz Y	are   Destructor	D d; d.gety d.gety return y is  or B() → D  or main() C c(5); c.sum() return 0  C c(5); return 0;	O; (); (); (); (); (); (); (); (); (); ()
int x; protey: protey: protey: public: int z; }  Class A{ private: int x; protected: int x; protected: int x; protected: int x; protected: int x; public: int ax;  public:	execution:    Class   Flat   Flat     Class   Flat     Class   Flat     Class   Flat   Flat     Class     Class   Flat     Class   Flat     Class   Flat     Cla	constructs  B: virtual B: virtual B: virtual Constructs  Class B Frivate: Class B Frivate: Fright Constructs  Class B Frivate: Fright Constructs  Class B Frivate: Fright	Class A ivate: ix; otected: ix;	Class D without vir without vir without vir ill be ambig ass A e: cted:	ala public of truct of truct of the content of the	class D:p class	are  oublic C { t accessible (){ cout< ''; (){ cout< ''; (){ cout or () t accessible	at main() C c(5); C.sum() return 0  at main() C sum() return 0  at main()	Destructor A.
Topic: [Con 1.Single Inh Class A {     private: int ax;     public: int;     public: int;     public: int; }  Coutreut:     int ax = a;     cout << endl;     int getax()     {         return a     }     cout << endl;     } };  OUTPUT: constructor A constructor A constructor B destructor B destructor A  Sequence of the  2. Multilevel  Class A {     private: int ax;     public:     A(int a)     {         ax = a;         cout << endl;     } };  OUTPUT: constructor B constructor B constructor B destructor A  Sequence of the  2. Multilevel  Class A {     return ax;     public:     A(int a)     {         ax = a;         cout << endl;     } };  OUTPUT: constructor B destructor A  Sequence of the  2. Multilevel  Class A {     private: int ax;     public:     A(int a)     {         ax = a;         cout << endl;     }     int getax()     {         return ax;     }      A()     {         cout << endl;     } };  OUTPUT: constructor A  Constructor A  Constructor B destructor A  Sequence of the  2. Multilevel  Class A {     private: int ax;     public: A     cout << endl;     }      A()     {         cout << endl;     }      cout << endl;     }      return ax; }  A'' << endl; }  Constructor A  Execution seque Destructor A  Execution seque Destructor B  Cosum = 15 D	itance:  class public {     // with will b     // will b  class B  class B     // will b  class	Constructed  B: virtual and virtual ambiguous and search and searc	Class A ivate:  Class A ivate:  Class A ivate:  Class B ivate:	ass A  in Inherit  without virial be ambig  structed:  y;  ass A  e:  tited: y;  tited:  ass A  e:  tited: y;  tited:  ass A  e:  tited: y;  tited: ass A  e:  tited: y;  tited: ass A  e:  tited: y;  tited: ass A  e:  tited: y;  tited: ass A  e:  tited: y;  tited: ass A  e:  tited:	all public of the public of th	sum() -  sible  class D:p  B, public  yoid gety  void gety  sty and  sty and  sty and  class D:p  B, public  yoid gety  sty and  sty and  sty and  class D:p  Roughlic  sty and  cla	Destructor  are  Destructor  are  Destructor  The property of	at main() C c(5); c sum(); return 0; return 0; return 0;	Destructor A.
F. Hybrid Inherical States in tax; public: int ax; public: int ax; public: int ax; public: int ax; public: int getax() {	itance:  class B class B dy and z are a eritance:  care constructor  destructor  are constructor  constructor  constructor  destructor  destructor  constructor  destructor  destructor  destructor  destructor  destructor  destructor	Construct  e:  Class B  private:  int bx  public:  class B  private:  int bx  public:  class B  private:  int bx  public:  and your  class B  private:  int bx:  your  class B:  private:  int bx:  y	Class A ivate:  ix ix:  ix ix:	Class D without vir without vi	al public of the	y and z  y and z  y and z  y and z  sible  class D:p  B, public  / x is no  void getz  v	are Destructor    Couldic   Court   C	at main() C c(5); C sum() T eturn 0  C c(5); C sum() T eturn 0;  T main() C c(5); T eturn 0;	Destructor A.
Topic: [Con int x; protectery int x; protectery int x; protectery int x; protected: int x; public: A (ax = a; cout <<< end; a (ax = a; cout << end;	intance:    class   public   will be   will be	Construct  Cass B:  Construct	Class A (and and and and and and and and and and	Class D without virial be ambig  ass A est; cted: y; and z are acal child class aberited from  in Inherit  structor  in Inherit  constructor  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; and z are acal child class aberited from  ass A est; cted: y; ass Class D ass Class D ass A est; cted: y; ass Class D ass A est; ass Class D ass A est; ass	al public of tall pub	sum() =  class D:  display and a  class D:  di	are    Diblic   C {   C   C   C   C   C   C   C   C	at main() C c(5); C sum(); return 0;  control C →  at main() C c(5); C sum(); C csum(); C csum(); C csum(); C csum(); C csum(); C control C →	Destructor B =
F. Hybrid Inherical int x; protected: int x; public: A(int a) { a caut < < < endl; } int getax() { cout < dendl; } int ax; public: and x; pub	itance:    class   public   { / with will b   };   will b   };   will b   };   will b   ;   will	constructs  e:  class B  for A'  class B	Class A ivate: service in typublic: service in typu	ass C: virturill be ambiguiting ass A  ass A	ala public of triuding and the content of the conte	ablic Bases D:  Class D:  By and Z  Sible Bases D:  Class D:  By ablic Bases D:  Class	and the content of th	at main() C c(5); c.sum(); return 0;  control C →  at main() C c(5); c.sum(); return 0;	Ostructor A.  Destructor B.  destructor B.  destructor C. →
5. Hybrid Inhered Service Serv	itance:    Class	Construct  e:  class B  for A'  class B	Class A in the control of the contro	ass C: virtue  ass A:	al public and public a	sum() =  sum	are	t main() C c(5); C.sum(); return 0;  at main() C c(5); C.sum(); return 0;  at main() C c(5); C.sum(); return 0;	estructor A.  Contractor A.  Contractor B.  Contrac
Topic: [Con 1.Single Inhered Class A {     private:     int ax;     public:     A (int a)     {         ax = a;         cout < ord         ax = a;         c	itance:   class   clas	Constructor  B: virtual  B: virtual  Find Dest  Class B:  Constructor  Find Dest  Class B:  Find Private:  Find	Class A  Class A  Class B  Cla	ass C: virtu  Class D  y and 2 are ac al child class ass A  est cted: y: tred  ass A  grad 2 are ac al child class ass A  est cted: y: tred  ass A  grad 2 are ac al child class ass A  est tred  y and 2 are ac al child class ass A  est tred  in Inheri  ass A  grad 2 are ac al child class ass A  est tred  in Inheri  ass A  est tred  in In	all public of a structor of a	r B'  r Garage	are Destructor    Control   Control	at main() C c(5); C.sum(); return 0;  at main() C c(5); C.sum(); return 0;  at main() B b(7); C.sum(); return 0;  at main() C c(5); c.sum(); return 0;	estructor A.  Contractor A.  Contractor B.  Contrac
Topic: [Con  S. Hybrid Inher  Class A  Frivate:  int x;  protected:  int x;  public:  A (int a)  A (ax = a;  Cout <<  Centl;  Freturn a  Bestructor A  constructor B  destructor B  destructor B  destructor B  contructor B  cont	itance:    class   cla	Constructor  e:  class B:  public:  B:  constructor  e:  class B:  public:  and private:  int bx  public:  and private:  int public:  and private:  and private:  int public:  and private:	Class A  A  A  A  Class A  A  A  A  A  A  A  A  A  A  A  A  A	ass A	class C: pub ala public   ala p	class D: cla	and the main (  and the main (	at main() C c(5); y is  at main() C c(5); C c(5); C c(5); C c(5); C csum(); Teturn 0;  at main() C c(5); Teturn 0;  at main() C c(5)	Destructor A.  ightharpoonup and the structor A.  ightharpoonup and the structor B.  i

// Name: Hasibul Hasan

Module-2

 $Git Hub\ Link: \underline{https://github.com/hasibul-hasib29/cse-1204/blob/master/module\_2.cpp$ 

// Roll: 2003127 Lab No - 4

#include<iostream>

using namespace std; class Test{ private: int x;