1. Operator ?: cannot be overloaded.

Operator += can be overloaded.

0.5 mark

with example 1.5 marks.
only can be overloaded= 0.5

2.

```
#include <iostream.h>
#include <iomanip.h>
class matrix
{
    int maxrow, maxcol;
    int * ptr;
public:
    matrix( int  r, int  c )
    {
        maxrow = r;
        maxcol = c;
        ptr    = new int [r * c];
    }
    void getmat( )
    {
        int i,j, mat_off,temp;
        cout << endl << "enter elements matrix:" << endl;
        for( i = 0; i < maxrow; i++ )
        {
            for( j = 0; j < maxcol; j++ )
            {
                mat_off = i * maxcol + j;
                cin >> ptr[ mat_off ];
            }
        }
    }
    void printmat( )
    {
        int i, j, mat_off;
        for( i = 0; i < maxrow; i++ )
        {
            cout << endl;
            for( j = 0; j < maxcol; j++ )
            {
                mat_off = i * maxcol + j;
                cout << setw( 3 ) << ptr[ mat_off ];
            }
        }
    }
    int delmat( )
    {
        matrix  q ( maxrow - 1, maxcol - 1 );
        int     sign  = 1, sum  = 0, i, j,k,count;
        int     newsize, newpos, pos, order;
        order = maxrow;
        if( order == 1 )
        {
            return ( ptr[ 0 ] );
        }
        for( i = 0; i < order; i++, sign *= -1 )
```

define highlight part properly
1 mark
otherwise as per your
understanding

0.5 mark

0.5 mark

```
    {
        for( j = 1; j < order; j++ )
        {
            for( k = 0, count = 0; k < order;k++ )
            {
                if( k == i )
                    continue;
                pos = j * order + k;
                newpos = ( j - 1 ) * ( order - 1 ) + count;
                q.ptr[ newpos ] = ptr[ pos ];
                count++;
            }
        }
        sum = sum + ptr[ i ] * sign * q.delmat( );
    }
    return ( sum );
}
matrix  operator +( matrix  b )
{
    matrix c ( maxrow, maxcol );
    int     i,j,mat_off;
    for( i = 0; i < maxrow; i++ )
    {
        for( j = 0; j < maxcol; j++ )
        {
            mat_off = i * maxcol + j;
            c.ptr[ mat_off ] = ptr[ mat_off ] + b.ptr[ mat_off ];
        }
    }
    return ( c );
}
matrix  operator *( matrix  b )
{
    matrix  c ( b.maxcol, maxrow );
    int     i,j,k,mat_off1, mat_off2, mat_off3;
    for( i = 0; i < c.maxrow; i++ )
    {
        for( j = 0; j < c.maxcol; j++ )
        {
            mat_off3 - i * c.maxcol + j;
            c.ptr[ mat_off3 ] = 0;
            for( k = 0; k < b.maxrow; k++ )
            {
                mat_off2 = k * b.maxcol + j;
                mat_off1 = i * maxcol + k;
                c.ptr[mat_off3]+=ptr[mat_off1]* b.ptr[mat_off2 ];
            }
        }
    }
    return ( c );
}
int operator ==( matrix  b )
{
    int i,j, mat_off;
    if( maxrow != b.maxrow || maxcol != b.maxcol )return ( 0 );
    for( i = 0; i < maxrow; i++ )
```

0.5 mark

1 mark

1 mark

```
        {
            for( j = 0; j < maxcol; j++ )
            {
                mat_off = i * maxcol + j;

                if( ptr[ mat_off ]!= b.ptr[ mat_off ] )
                     return ( 0 );
            }
        }
        return ( 1 );
    }
}    ;

void main( )
{
    int rowa, cola, rowb, colb;
    cout << endl << "Enter dimensions of matrix A";
    cin >> rowa >> cola;
    matrix a ( rowa, cola);
    a.getmat( );
    cout << endl << "Enter dimensions of matrix B";
    cin >> rowb >> colb;
    matrix b ( rowb, colb);
    b.getmat( );
    matrix  c ( rowa, cola);
    c = a + b;
    cout << endl << "The sum of two matrics = ";
    c.printmat( );
    matrix  d ( rowa, colb );
    d = a * b;
    cout << endl << "The product of two matrics = ";
    d.printmat( );
    cout << endl << "Determinant of matrix a =" << a.delmat( );
    if( a == b )
        cout << endl << "a & b are equal";
        else
        cout << endl << "a & b are not equal";
}
```

0.5

3.

```
#include<iostream.h>
   #include<conio.h>
class complex
  {
        float x;
        float y;
  public:
        void input(float real, float img)
        {
              x=real;
              y=img;
        }
        friend complex sum(complex, complex);
```

define friend function properly and create class = 1 mark

```
        void show(complex);
};
complex sum(complex c1, complex c2)
{
        complex c3;
        c3.x = c1.x + c2.x;
        c3.y   =   c1.y   +   c2.y;
        return (c3);
}
void complex :: show(complex c)
{
        cout<<c.x<<"+j"<<c.y<<"\n";
}
int main()
{
        complex A,B,C;
        A.input(3.1, 5.65);
        B.input(2.75, 1.2);

        C=sum(A,B);
        cout<<"A=";
        A.show(A);
        cout<<"B=";
        B.show(B);
        cout<<"C=";
        C.show(C);

        return 0;
}
```

0.5

0.5

1 mark

4. **A.**

```
int* getCountAddress ( )
{
return &count;
}
```

Note: All other code remain unchanged

each part 1.5 marks

B. Just erase “objRoom4 = objRoom3; invalid to call copy constructor.” for successfully run.

C. Solution: setValue(int w, int h) method should be public.

D. The argument of Space() function is void type, so when this function is called there are no argument can send to it. But ‘mCount’ argument is sending to Space() function through return
space(mCount); Statement.
Here return space (mCount); replaced by return space();

5. A. LINE1 a=x; 0.5 marks

LINE2 b=y; 0.5 marks

LINE3 public: 1 mark

B. LINE1 complex(complex &c) 1 mark

LINE2 b=c.b; 0.5 mark

LINE3 a=c.a; 0.5 mark