The following programs have been successfully compiled & run without any warnings or errors on GNU C++ version 3.2.

```
// 1. Largest & Smallest

#include<iostream>
using namespace std;

int main()
{
 float n1,n2,n3;

 cout<<"Enter three numbers"<<endl;
 cin>>n1>>n2>>n3;
 if(n1>=n2)
                {
                                if(n1>=n3)
                                { cout<<n1<<" is the largest number" <<endl;
                                                if(n2>=n3)cout<<n3 <<" is the smallest
number";
                                                else cout<<n2 <<" is the largest number";
                                }
                                else
                                  {
                                                cout<<n3<<" is the largest number" <<endl;
                                                cout<<n2<<" is the smallest number";
                                  }
                }
else{
 if(n2>=n3)
                {
                                cout<<n2<<" is the largest number" <<endl;
                                if(n3>=n1)cout<<n1 <<" is the smallest number";
                                else cout<<n3 <<" is the smallest number";
                }
else
                {
                  cout<<n3<<" is the largest number";
                                if(n1>=n2)cout<<n2 <<" is the smallest number" <<endl;
                                else cout<<n1 <<" is the smallest number";
                }
                                }
cout<<endl;
cout<<endl;
return 0;

}
```

// 2. Program to read a line of text & reverse that using recursion.

```
#include<iostream>
using namespace std;

char a[50];
char c;
int i=0;
main()
{

void printline();

cout<<"Enter a line of text (max length - 50 characters)"<<endl;

printline();

cout<<endl;

return 0;
}


void printline()
{

cin.get(c);
a[i]=c;
i++;
if(c != '\n')printline();

i--;
cout<<a[i];

}
```

```
// 3. Class of complex numbers

#include<iostream>
using namespace std;

class complex
{
float i,r;
public :
                void getdata();
                void putdata();
                complex add(complex c);
};

complex complex::add(complex c)
{
complex t;
t.i=i+c.i;
t.r=r+c.r;
return t;
};

void complex::getdata()
{
cout<<"Enter value for integer part    ";
cin>>i;
cout<<"Enter value for the real part   ";
cin>>r;
};

void complex::putdata()
{
cout<<endl<<i<<" + i"<<r;
};

int main()
{
complex x1,x2,x3;
x1.getdata();
x2.getdata();
x3=x1.add(x2);
x3.putdata();
cout<<endl;
return 0;
}
```

```
// 4. Constructor Overloading

#include<iostream>
using namespace std;

class arith
{
float a,b;

public :
                arith();
                arith(float m);
                arith(float m, float n);
                void prn();
};

arith::arith()
{
                a = 10;
                b = 50;
};

arith::arith(float m)
{
                a = m;
                b = 0;
};

arith::arith(float m, float n)
{
                a = m;
                b = n;
};

void arith::prn()
{
                cout<<endl<<a;
                cout<<endl<<b;
};

int main()
{

arith a1;
arith a2(10);
arith a3(50,30);

a1.prn();                       //10,50
```

```
cout<<endl;
a2.prn();                    //10,0
cout<<endl;
a3.prn();                    //50,30

cout<<endl;
return 0;
}
```

```
// 5. Default Argument

#include<iostream>
using namespace std;

class arith
{
float a,b;

public :
                arith(float m, float n = 0);
                void prn();
};

arith::arith(float m, float n)
{
a = m;
b = n;
};

void arith::prn()
{
cout<<endl<<a;
cout<<endl<<b;
};

int main()
{
arith a2(10);
arith a3(40,30);

                cout<<endl;
                a2.prn();                       //10,0
                cout<<endl;
                a3.prn();                       //40,30
                cout<<endl;
return 0;
}
```

```
// 6. Dynamic initialisation of constructor

#include<iostream>
using namespace std;

class constr
{

int a,b;

public :
                constr(int a1, int b1)
                {
                   a = a1;
                   b = b1;
                }
                void print();
};

int main()
{
int x,y;

                cout<<"Enter x \t";
                cin>>x;
                cout<<"Enter y \t";
                cin>>y;

constr c(x,y);
c.print();

return 0;
}

void constr::print()
{
cout<<"\n"<<a<<"\t"<<b;
cout<<endl;

}
```

```
// 7. Matrix manipulations

#include<iostream>
using namespace std;

const int row=2,col=2;

class matrix
{
protected :
                int mat[row][col];

public :
                void inputmatrix();
                void displaymatrix();
                matrix mul(matrix m3,matrix m4);
                matrix add(matrix m3,matrix m4);
                matrix sub(matrix m3,matrix m4);
                matrix transpose(matrix m);

};


int main()
{
matrix m1,m2,mm,t;

m1.inputmatrix();
m2.inputmatrix();

                cout<<endl;
                cout<<"Matrix 1 "<<endl;
                m1.displaymatrix();
                cout<<"Matrix 2 "<<endl;
                m2.displaymatrix();

cout<<endl;

cout<<"Sum of the two matrices  "<<endl;
mm=t.add(m1,m2);
mm.displaymatrix();
cout<<"Difference of the two matrices  "<<endl;
mm=t.sub(m1,m2);
mm.displaymatrix();

cout<<"Transpose of Matrix 1 "<<endl;
mm=t.transpose(m1);
mm.displaymatrix();
```

```
cout<<"Transpose of Matrix 2 "<<endl;
mm=t.transpose(m2);
mm.displaymatrix();

cout<<"Product of the two matrices  "<<endl;
mm=t.mul(m1,m2);
mm.displaymatrix();

cout<<endl;
return 0;
};


void matrix::inputmatrix()
{
for(int i=0; i<row; i++)
{
for(int j=0; j<col; j++)
{
cout<<"Enter  element "<<i<<j<<"    ";
cin>>mat[i][j];
}
}
};


void matrix::displaymatrix()
{
cout<<endl;
for(int i=0; i<row; i++)
{
for(int j=0; j<col; j++)
{
cout<<"   "<<mat[i][j]<<"    ";
}
cout<<endl;
}
};


matrix matrix::add(matrix m3,matrix m4)
{
matrix temp;

for(int i=0; i<row; i++)
for(int j=0; j<col; j++)
temp.mat[i][j]=m3.mat[i][j]+m4.mat[i][j];
return temp;
```

```
};


matrix matrix::sub(matrix m3,matrix m4)
{
matrix temp;

for(int i=0; i<row; i++)
for(int j=0; j<col; j++)
temp.mat[i][j]=m3.mat[i][j]-m4.mat[i][j];
return temp;

};

matrix matrix::transpose(matrix m)
{
matrix temp;

for(int i=0; i<row; i++)
for(int j=0; j<col; j++)
temp.mat[j][i]=m.mat[i][j];
return temp;


};

matrix matrix::mul(matrix m3,matrix m4)
{
matrix temp;

for(int i=0; i<row; i++)
for(int j=0; j<col; j++)
for(int k=0; k<row; k++)
temp.mat[i][j]=m3.mat[i][k]*m4.mat[k][j];

return temp;
};
```

```
// 8.Creating 1-d array at run time

#include<iostream>
using namespace std;

int main()
{
int size, *a;

cout<<"Enter array size\n";
cin>>size;

a= new int[size]; //Now a is an array of size size.

for(int i=0; i<size; i++)
{ cout<<"Enter value for "<<i+1<<" element \t";
  cin>>a[i];
}


for(int i=0; i<size; i++)
{
cout<<"The value at "<<i+1<<" array location is \t"<<a[i]<<endl;
}
cout<<endl;
return 0;
}
```

```
// 9. Creating a 2-d array dynamically

#include<iostream>
using namespace std;

int main()
{
int **a,row,col;

cout<<"Enter row size \t";
cin>>row;
cout<<"Enter column size\t";
cin>>col;

a = new int *[row];
for(int i=0; i<row; i++)
{
                a[i] = new int [col];
}

for(int i=0; i<row; i++)
for(int j=0; j<col; j++)
{
cout<<"Enter "<<i<<" "<<j<<" element\t";
cin>>a[i][j];
}

for(int i=0; i<row; i++)
{
for(int j=0; j<col; j++)
{
cout<<a[i][j];
cout<<"\t";
}
cout<<endl;
}
cout<<endl;
return 0;
}
```

```
// 10. Stack

#include<iostream>
using namespace std;

struct node
{
int data;
node *next;
};

class stk
{
  node *top;

public :
stk() {top =NULL;}
void pop();
void push(int r);
void traverse();

};


int main()
{
stk l;
char ch;
int t,i;

do{
cout<<"Enter the no. you wanna push\t";
cin>>i;
l.push(i);
cout<<"Wanna add another no. : y / n    ";
cin>>ch;

}while((ch=='y')||(ch=='Y'));


cout<<"Wanna pop any value  y / n     ";
cin>>ch;
if((ch=='y')||(ch=='Y'))
{

cout<<"How many nos. you wanna pop";
cin>>t;
for(i=0; i<t; i++)l.pop();
```

```
};

cout<<endl;
cout<<"The current stack contains :";
cout<<endl;
l.traverse();


cout<<endl;
return 0;
}


void stk::push(int r)
{
node *temp, *topn;

temp=new node;
temp->data=r;
temp->next=NULL;

if(top==NULL)top=temp;
else
{
temp->next=top;
top=temp;
}

};



void stk::pop()
{
node *p;

p=top;

top=top->next;

cout<<endl<<p->data<<endl;

delete p;

cout<<endl;
};
```

```
void stk::traverse()
{
node *t=top;

if(t==NULL)cout<<"The current stack is empty";
while(t!=NULL)

{
 cout<<t->data;
t=t->next;
cout<<endl;

}


};
```

```
// 11. Queue

#include<iostream>
using namespace std;

struct node
{
int data;
node *next;
};

class que
{
		node *front, *back;
public :
		que() {front =NULL; back = NULL; }
		void rem();
		void create();
		void traverse();
};

int main()
{
que l;
int loop;

		cout<<"Enter how many nodes do you wanna create\t";
		cin>>loop;
for(int i=0; i<loop; i++)
{
l.create();
}
l.rem();
l.traverse();

cout<<endl;
return 0;
}

void que::create()
{
node *temp;

temp=new node;
cout<<"Enter data\t";
cin>>temp->data;
temp->next=NULL;
```

```
if(front==NULL)
{
front=temp;
back=temp;
}
else
{
back->next=temp;
back=temp;
}
}

void que::rem()
{
int x,w=0;
node *p=front;

front=front->next;
cout<<endl<<p->data<<"  removed from the queue "<<endl;

delete p;

cout<<endl;
}

void que::traverse()
{
node *t=front;
cout<<"The remaining elements in the queue are : "<<endl;

while(t!=NULL)

{
 cout<<t->data;
t=t->next;
cout<<endl;

}
}
```

```
// 12. Link list

#include<iostream>
using namespace std;

struct node
{
int data;
node *next;
};

class link
{
  node *root;

public :
link() {root =NULL;}
void remove();
void create();
void traverse();

};


int main()
{
link l;
int loop;

cout<<"Enter how many nodes do you wanna create\t";
cin>>loop;

for(int i=0; i<loop; i++)
{
l.create();
}
l.remove();
l.traverse();


cout<<endl;
return 0;
}


void link::create()
{
node *temp, *rootn;
```

```
temp=new node;
cout<<"Enter data\t";
cin>>temp->data;
temp->next=NULL;

if(root==NULL)root=temp;
else
{
rootn=root;
while(rootn->next!=NULL)
rootn=rootn->next;
rootn->next=temp;


}


}

void link::remove()
{
int x,w=0;
node *b,*p;
cout<<"Enter the value to be deleted\t";
cin>>x;

b=NULL;
p=root;

while(p->next!=NULL)
{
if((p->data==x)||(p->next->data==x))w++;
p=p->next;
}

p=root;

if(w!=0)
{
while((p->data!=x)&&(p!=NULL))
{
b=p;
p=p->next;
}
}

if(w==0)cout<<"element doesn't exist in the current link list";
```

```
else
{
if(b==NULL)root=root->next;
else
b->next=p->next;
delete p;
}
cout<<endl;
}


void link::traverse()
{
node *t=root;

while(t!=NULL)

{
 cout<<t->data;
t=t->next;
cout<<endl;

}



}
```

```
// 13. Single Level Inheritance

#include<iostream>
using namespace std;

class mathe
{
protected : int a,b;

public :

                long add();
                long sub();
                long mul();
                void getdata();
};

class onemore:public mathe
{
public :
                double div();

};


int main()
{

onemore o1;

o1.getdata();

cout<<"The addition       of the entered numbers is  :  "<<o1.add()<<endl;
cout<<"The subtraction    of the entered numbers is  :  "<<o1.sub()<<endl;
cout<<"The multiplication of the entered numbers is  :  "<<o1.mul()<<endl;
cout<<"The division       of the entered numbers is  :  "<<o1.div()<<endl;


return 0;
};

void mathe::getdata()
{
cout<<"Enter the first  integers   ";
cin>>a;
cout<<"Enter the second integer    ";
cin>>b;
};
```

```
long mathe::add()
{
return a+b;
};

long mathe::sub()
{
return a-b;
};

long mathe::mul()
{
return a*b;
};

double onemore::div()
{
return double(a)/b;
};
```

```
// 14. Multilevel Inheritance

#include<iostream>
using namespace std;

class vehicle
{
protected :
                int maxspeed;

public :
                vehicle()
                {cout<<"Vehicles have started moving"<<endl;};
};

class car:public vehicle
{
protected :
                char modelno[10];

public :
                car()
                {cout<<"Who's in this car"<<endl;};
};

class brand:public car
{
protected :
char company[20];

public :
                void putdata();
                void getdata();
};


int main()
{
brand bebe;

bebe.getdata();
bebe.putdata();

return 0;
};


void brand::getdata()
```

```
{
cout<<"Enter the car's maxspeed        ";
cin>>maxspeed;
cout<<"Enter car's model nuumber       ";
cin>>modelno;
cout<<"Enter the manufacturer's name   ";
cin>>company;

};

void brand::putdata()
{
cout<<"The details of the car are as below :"<<endl<<endl;
cout<<"Max Speed    =   "<<maxspeed<<endl;
cout<<"Model number =   "<<modelno<<endl;
cout<<"Manufacturer =   "<<company<<endl;

};
```

```
// 15. Multiple Inheritance

#include<iostream>
using namespace std;

class locks
{
protected :
                int wt;
                char pcode[15];

};

class alarms
{
protected :
                char color[10];
                int cost;
};

class security:public locks, public alarms
{
protected :
                char device[10];

public :
                void getdata();
                void putdata();
};


int main()
{
security ss;

ss.getdata();
ss.putdata();

return 0;
};


void security::getdata()
{
cout<<"Enter the devices to be used for security  (write l for locks or a for
alarms)   ";
cin>>device;
if(device[0]=='l')
```

```
{
cout<<"Enter the weight of lock   ";
cin>>wt;
cout<<"Enter the product code     ";
cin>>pcode;
cout<<"Enter the cost  ";
cin>>cost;

}
else
{
cout<<"Enter the color of the alarm  ";
cin>>color;
cout<<"Enter the cost  of the alarm  ";
cin>>cost;
};

};

void security::putdata()
{
if(device[0]=='l')
{
cout<<endl<<wt<<endl<<pcode<<endl<<cost<<endl;
}
else
{
cout<<endl<<color<<endl<<cost<<endl;
};


};
```

```
// 16. Pure Virtual function

#include<iostream>
using namespace std;

class base
{
public :
virtual void show()=0;
};

class derived1:public base
{
public :
void show()
{
cout<<endl<<"Derived 1";
};
};

class derived2:public base
{
public :
void show()
{
cout<<endl<<"Derived 2";
};
};

int main()
{
base *pp[2];
derived1 dd;
derived2 ddd;

pp[0]=&dd;
pp[1]=&ddd;

pp[0]->show();          //Derived 1
pp[1]->show();          //Derived 2

cout<<endl;
return 0;
};
```

```
// 17. Friend function

#include<iostream>
using namespace std;

class example2;
class example1
{
int a;
public :
                void getdata();
                friend long add(example1 e1, example2 e2);
};

class example2
{
int b;
public :
                void getdata();
                friend long add(example1 e1, example2 e2);
};

long add(example1 e1, example2 e2)
{
int r;
r=e1.a+e2.b;
return r;

};

int main()
{
example1 ee;
example2 edd;

ee.getdata();
edd.getdata();

cout<<endl<<add(ee,edd)<<"   is the sum "<<endl;

return 0;
};


void example1::getdata()
{
cout<<endl<<"Enter first no.  ";
cin>>a;
```

```
};

void example2::getdata()
{
cout<<endl<<"Enter the second no.  ";
cin>>b;
};
```

```
// 18. Friend Classes

#include<iostream>
using namespace std;

class pal
{
                friend class buddy;
private :
                int b;
public :
                pal()
                {b=9;};
};

class buddy
{
                pal pp;
public :
                void func1()
                {
                cout<<endl<<"value =   "<<pp.b;
                pp.b++;
                 };
                void func2()
                {
                cout<<endl<<"value =   "<<pp.b;
                pp.b*=pp.b;
                };
                void func3()
                {
                 cout<<endl<<"value =   "<<pp.b;
                };
};

int main()
{
buddy bb;

                bb.func1();
                bb.func2();
                bb.func3();
cout<<endl;
return 0;
};
```

// 19. Use of static data member function within a class

```
#include<iostream>
using namespace std;

class exa
{
        int a;
        static int b;
public :
        void getdata();
        void putdata();
};

int exa::b;     //initialisation statement

int main()
{
exa e1,e2,e3,e4;

        e1.putdata();           //1108517584,0
        e2.getdata();
        e2.putdata();           //5,6
        e3.putdata();           //134513982,6
        e4.getdata();
        e4.putdata();           //2,3;
return 0;
};

void exa::getdata()
{
cout<<"Enter value for a   ";
cin>>a;
cout<<"Enter value for b   ";
cin>>b;

};

void exa::putdata()
{
cout<<"a  =  "<<a<<endl;
cout<<"b  =  "<<b<<endl;

};
```

```
// 20. Static function member within a class

#include<iostream>
using namespace std;

class freak
{
                static int total;
                int id;
public :
freak()
  {
                total++;
                id=total;
  };
~freak()
  {
                total--;
                cout<<endl<<"Destroyed id no.   "<<id<<endl;
  };
static void showtotal()
  {
                cout<<endl<<"The total is   "<<total<<endl;
  };
void showid()
  {
                cout<<endl<<"Id no. is   "<<id<<endl;
  };
};
int freak::total;

int main()
{
freak ff;

freak::showtotal();

freak ff1,ff2;
freak::showtotal();
ff.showid();
ff1.showid();
ff2.showid();

return 0;
};
```

// 21. Type Conversion (Class to basic conversion)

```
#include<iostream>
using namespace std;

class dist
{
float i,f;
public :
                void getdata();

                operator float()
                {
                  float t;
                  t=f/12;
                  t=t+i;
                  return t/3.28;
                }
};


void dist::getdata()
{
cout<<"Enter value for i    ";
cin>>i;
cout<<"Enter value for f    ";
cin>>f;

};

int main()
{

float mt;
dist d1;

d1.getdata();
mt=d1;                          //also we can write : mt=float(d1);

cout<<"Toatl distance is  "<<mt<<" meters";

cout<<endl;
return 0;
}
```

// 22. Type Conversion (Basic to class conversion)

```
#include<iostream>
using namespace std;

class dist
{
float i,f;
public :
                void putdata();
                dist(float m)
                {
                                float x;
                                int t;
                                x=3.28*m;
                                f=int(x);
                                i=12*(x-f);
                };
};

void dist::putdata()
{
cout<<"inches = "<<i<<endl;
cout<<"feet   = "<<f<<endl;
};

int main()
{
float mt;
cout<<"Enter distance in meters   ";
cin>>mt;

dist d(mt);

d.putdata();

cout<<endl;
return 0;
};
```

// 23. Type conversion(class to class; function in source class)

```
#include<iostream>
#include<math.h>
using namespace std;

class cart
{
private : float x,y;
public :
                cart()
{
x=0;
y=0;
};

                cart(float xx, float yy)
{
x=xx;
y=yy;
};

                void putdata()
{
cout<<"The x coordinate is   "<<x<<endl;
cout<<"The y coordinate is   "<<y<<endl;
};
};

class polar
{
private: float ang,rad;
public :
                void getdata()
{
cout<<"Enter angle   ";
cin>>ang;
cout<<"Enter radius  ";
cin>>rad;
};
                operator cart()
{
float xx=rad*cos(ang);
float yy=rad*sin(ang);
return cart(xx,yy);
};

};
```

```
int main()
{
cart c1;
polar p1;
p1.getdata();
c1=p1;

c1.putdata();
return 0;
};
```

// 24. Type conversion(class to class; function in destination class)

```
#include<iostream>
#include<math.h>
using namespace std;

class polar
{
private: float ang,rad;
public :
                void getdata()
{
cout<<"Enter angle   ";
cin>>ang;
cout<<"Enter radius  ";
cin>>rad;
};

float getrad()
{
return rad;
};

float getang()
{
return ang;
};

};


class cart
{
private : float x,y;
public :
                cart()
{
x=0;
y=0;
};


                void putdata()
{
cout<<"The x coordinate is   "<<x<<endl;
cout<<"The y coordinate is   "<<y<<endl;
};
```

```
                cart(polar p)
{
float r=p.getrad();
float a=p.getang();
x=r*cos(a);
y=r*sin(a);
};

};

int main()
{
cart c1;
polar p1;
p1.getdata();
c1=p1;

c1.putdata();
return 0;
};
```

```
// 25. Overloading of assignment operator

#include<iostream>
using namespace std;

class complex
{
float i,r;
public:
                void getdata();
                void putdata();
                void operator +=(complex c);
};
void complex::operator +=(complex c)
{
i+=c.i;
r+=c.r;
};
void complex::getdata()
{
cout<<"Enter value for integer part    ";
cin>>i;
cout<<"Enter value for the real part   ";
cin>>r;
};
void complex::putdata()
{
cout<<i<<" + i"<<r;
};

int main()
{

complex x1,x2;

x1.getdata();
x2.getdata();

x2+=x1;

x2.putdata();

cout<<endl;
return 0;
};
```

```
// 26. class complex with overloading of the binary operator

#include<iostream>
using namespace std;

class complex
{
float i,r;
public:
                void getdata();
                void putdata();
                complex operator +(complex c);
};

complex complex::operator +(complex c)
{
                complex t;
                t.i=i+c.i;
                t.r=r+c.r;
                return t;
};

void complex::getdata()
{
cout<<"Enter value for integer part    ";
cin>>i;
cout<<"Enter value for the real part   ";
cin>>r;
};

void complex::putdata()
{
cout<<endl<<i<<" + i"<<r;
};

int main()
{
complex x1,x2,x3;
                x1.getdata();
                x2.getdata();
                x3=x1+x2;
                x3.putdata();
cout<<endl;
return 0;
};
```

```
// 27. post increment by overloading unary operator

#include<iostream>
using namespace std;

class complex
{
float i,r;

public:
                void getdata();
                void putdata();
                void operator ++(int);
};
void complex::operator ++(int)
{
++r;
i++;
};

void complex::getdata()
{
cout<<"Enter value for integer part    ";
cin>>i;
cout<<"Enter value for the real part   ";
cin>>r;
};
void complex::putdata()
{

cout<<i<<" + i"<<r;
};

int main()
{
complex x1;

x1.getdata();
x1++;
x1.putdata();

cout<<endl;
return 0;
};
```

```
// 28. pre increment by overloading unary operator

#include<iostream>
using namespace std;

class complex
{
float i,r;

public:
                void getdata();
                void putdata();
                void operator ++();
};

void complex::operator ++()
{
++r;
i++;
};

void complex::getdata()
{
cout<<"Enter value for integer part    ";
cin>>i;
cout<<"Enter value for the real part   ";
cin>>r;
};

void complex::putdata()
{

cout<<i<<" + i"<<r;
};

int main()
{
complex x1;
x1.getdata();
++x1;
x1.putdata();
cout<<endl;
return 0;
};
```

```
// 29. Overloading of relational operator

#include<iostream>
using namespace std;

class complex
{
float i,r;

public:
                void getdata();
                bool operator <=(complex c);
};

bool complex::operator <=(complex c)
{
float l1,l2;
l1=sqrt(i*i+r*r);
l2=sqrt(c.i*c.i+c.r*c.r);
if(l1<=l2)return true;
else return false;
};

void complex::getdata()
{
cout<<"Enter value for integer part    ";
cin>>i;
cout<<"Enter value for the real part   ";
cin>>r;
};

int main()
{

complex x1,x2,x3;

x1.getdata();
x2.getdata();

if(x1<=x2)cout<<"X1 is less than or equal to X2";
else cout<<"X1 is greater than X2";

cout<<endl;
return 0;
};
```

// 30. read n characters & save them to file

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
char c[3500];
int n;

fstream f;
f.open("abc.txt",ios::out);

cout<<"Enter the paragraph(s) size "<<endl;
cin>>n;

if(n<3500)
{
cout<<"Type your words  & press   $   to stop "<<endl;
cin.get(c,n,'$');

cout<<endl<<"You entered :"<<endl<<c;


f.write((char *)&c,n);

};


f.close();
cout<<endl;
return 0;

};
```

// 31. find out the size of a file

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
int n;
fstream f;
f.open("copy.txt",ios::in);

f.seekg(0,ios::end);
n=f.tellg();

cout<<"The file size is    "<<n;

f.close();
cout<<endl;
return 0;
};
```

// 32. read a file & print it on the screen

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
char c;
fstream f;
f.open("abc.txt",ios::in);

while(f)
{
f.get(c);
cout<<c;
};

f.close();
cout<<endl;
return 0;
};
```

// 33. read n characters from a file & print them on the screen

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
char c[1500];
int n;
fstream f;
f.open("abc.txt",ios::in);

cout<<"Enter how many characters you want to read   ";
cin>>n;

f.read((char *)&c,n);

cout<<endl<<c;

f.close();
cout<<endl;
return 0;
};
```

// 34. read n characters from a file & save them to another file

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
char c[1500];
int n;
fstream f;
f.open("abc.txt",ios::in);

cout<<"Enter how many characters you want to read   ";
cin>>n;

f.read((char *)&c,n);

f.close();
cout<<endl;

f.open("copy.txt",ios::out);
f.write((char *)&c,n);

f.close();
cout<<endl;

return 0;
};
```

// 35. Read 2 files & store them to a third file

```
#include<fstream>
#include<iostream>
using namespace std;

int main()
{
char c[1500];
int n,s=0;
fstream f1,f2;
f1.open("abc.txt",ios::in);
f2.open("add.txt",ios::out);

f1.seekg(0,ios::end);
n=f1.tellg();
cout<<"The first file size is    "<<n;
s=s+n;

f1.seekg(0,ios::beg);
f1.read((char *)&c,n);
f1.close();
cout<<endl;

f2.write((char *)&c,n);
f1.open("copy.txt",ios::in);
f1.seekg(0,ios::end);
n=f1.tellg();
cout<<"The second file size is    "<<n;
s=s+n;

f1.seekg(0,ios::beg);
f1.read((char *)&c,n);
f1.close();
cout<<endl;

f2.write((char *)&c,n);
f2.close();

cout<<"The output filesize is   "<<s;
cout<<endl;

return 0;
};
```