```
//Pointer as a data member of class
#include<iostream>
using namespace std;
class Array
{
int *arr;
int size;
public:
void get_data(int n)
{
size=n;
arr=new int[size];
cout<<"\nEnter elements:";
for(int i=0;i<size;i++)
{
cin>>*(arr+i);
}
}
void add()
{
int sum=0;
for(int i=0;i<size;i++)
{
sum+=*(arr+i);
}
cout<<"\n Sum of elements="<<sum;
delete []arr;//Memory deallocation
cout<<"\nMemory deallocated";
}
};
int main()
{
Array a;
int n;
cout<<"\n Enter the number of elements:"<<endl;
cin>>n;
a.get_data(n);
a.add();
return 0;
}
```

```
//Pointer as a data member of class
#include<iostream>
using namespace std;
class Array
{
int *arr;
int size;
public:
void get_data(int n)
```

```
{
size=n;
arr=new int[size];
cout<<"\nEnter elements:";
for(int i=0;i<size;i++)
{
cin>>*(arr+i);
}
}
int largest()
{
int max=*(arr+0);
for(int i=1;i<size;i++)
{
if(*(arr+i)>max)
{
   max=*(arr+i);
}
}
delete []arr;//Memory deallocation
return max;
}
};
int main()
{
Array a;
int n;
cout<<"\n Enter the number of elements:"<<endl;
cin>>n;
a.get_data(n);
cout<<"\nLargest element from the given array elements is:"<<a.largest();
return 0;
}
//Pointer to object-Example 1
#include<iostream>
using namespace std;
class A
{
int x;
public:
void getdata()
{
cout<<"\n Enter value for x:"<<endl;
cin>>x;
}
void showdata()
{
cout<<"\n Entered value is:"<<x<<endl;
}
```

```
};
int main()
{
A obj1;
A *ptr;
ptr=&obj1;//Pointer to object
ptr->getdata();
ptr->showdata();
(*ptr).getdata();
(*ptr).showdata();
}
//Pointer to object-Example 2
#include<iostream>
using namespace std;
class example
{
int x;
public:
void getdata(int a)
{
x=a;
}
void show()
{
cout<<"Value of x:"<<x<<"\n";
}
};
int main()
{
example *p=new example[2];
example *d=p;
example *flag=p;
int a;
for(int i=0;i<2;i++)
{
cout<<"\nEnter value:";
cin>>a;
p->getdata(a);
p++;
}
for(int i=0;i<2;i++)
{
cout<<"Value is:"<<"\n";
d->show();
d++;
}
delete[]flag;//Deallocation of memory
return 0;
}
```

```
//Pointer to object-Example 2
#include<iostream>
using namespace std;
class rectangle
{
int l,b;
public:
void input()
{
cout<<"\nEnter values of length and breadth:";
cin>>l>>b;
}
int area()
{
return l*b;
}
};
int main()
{
int n;
cout<<"\nEnter number of rectangles you want to work with:";
cin>>n;
rectangle *p=new rectangle[n];
rectangle *d=p;
rectangle *flag=p;
for(int i=0;i<n;i++)
{
p->input();
p++;
}
for(int i=0;i<n;i++)
{
cout<<"Area is:";
cout<<d->area()<<endl;
d++;
}
delete[]flag;//Deallocation of memory
return 0;
}
//Pointer to data member
#include<iostream>
using namespace std;
class Test
{
public:
int x;
void show_data();
};
void Test::show_data()
```

```
{
cout<<"\n x="<<x;
}
int main()
{
Test t;
int Test::*ptr=&Test::x;
//int Test::*ptr;
//ptr=&Test::x;
t.*ptr=20;//.* is member dereferencing operator
t.show_data();
Test *tp=new Test;//Dynamically allocate memory for object
tp->*ptr=80;
tp->show_data();
(*tp).*ptr=70;
(*tp).show_data();
}
//Pointer to data member
#include<iostream>
using namespace std;
class Test
{
public:
int x;
void input()
{
   cout<<"\nEnter x:";
   cin>>x;
}
};
int main()
{
Test t;
int Test::*ptr=&Test::x;
t.input();
cout<<"\nPerimeter of square is:"<<4*(t.*ptr);
Test *tp=new Test;//Dynamically allocate memory for object
tp->input();
cout<<"\nPerimeter of square is:"<<4*(tp->*ptr);
}

//Pointer to member function
#include<iostream>
using namespace std;
class Test
{
public:
void show_msg(int);
};
```

```
void Test::show_msg(int x)
{
cout<<"\n Hello World!!!"<<x;
}
int main()
{
void(Test::*fp)(int)=&Test::show_msg;
//Return type of function is void and it is not accepting any argument
Test t;
(t.*fp)(5);//.*(Member dereferencing operator
Test *ptr=new Test;
(ptr->*fp)(5);
}
//Pointer to member function
#include<iostream>
using namespace std;
class Test
{
public:
int x;
void input()
{
    cout<<"\nEnter x:";
    cin>>x;
}
};
int main()
{
Test t;
int Test::*ptr=&Test::x;
void(Test::*fp)()=&Test::input;
(t.*fp)();
cout<<"\nPerimeter of square is:"<<4*(t.*ptr);
Test *tp=new Test;//Dynamically allocate memory for object
(tp->*fp)();
cout<<"\nPerimeter of square is:"<<4*(tp->*ptr);
}
//When local variable's name is same as member's name
#include<iostream>
using namespace std;
class Test
{
   int x;
public:
   void setX (int x)
   {
       this->x = x;
   }
   void print()
   {
    cout << "x = " << x << endl;
    cout<<"Address of current object:"<<this<<endl;
   }
};

int main()
{
   Test obj,obj1,obj2;
   int x=20;
   int x1=30;
   int x2=40;
   obj.setX(x);
   obj.print();
   obj1.setX(x1);
   obj1.print();
   obj2.setX(x2);
   obj2.print();
   return 0;
}
#include<iostream>
using namespace std;
class Test
{
private:
int x;
int y;
public:
void assign(int x, int y)
{
    this->x = x;
    this->y = y;
}
Test &setX(int a)
{
     x = a;
     return *this;
}
Test &setY(int b)
{
    y = b;
    return *this;
}
void print()
{
    cout << "x = " << x << " y = " << y << endl;
}
};
int main()
```

```
{
Test obj1;
obj1.assign(5,5);
// Chained function calls. All calls modify the same object as the same object is returned by reference
obj1.setX(10).setY(20).print();
return 0;
}
```