**Assignment – 28**

**Operator Overloading, friend operator and this pointers**

**1.** **#include <iostream>**

**using namespace std;**

**class Complex**

**{**

**private:**

**int real, imag;**

**public:**

**Complex(int r = 0, int i =0)**

**{ real = r; imag = i; }**

**friend ostream & operator << (ostream &out, const Complex &c);**

**friend istream & operator >> (istream &in, Complex &c);**

**};**

**ostream & operator << (ostream &out, const Complex &c)**

**{**

**out << c.real;**

**out << "+i" << c.imag << endl;**

**return out;**

**}**

**istream & operator >> (istream &in, Complex &c)**

**{**

**cout << "Enter Real Part ";**

**in >> c.real;**

**cout << "Enter Imaginary Part ";**

**in >> c.imag;**

**return in;**

**}**

**int main()**

**{**

**Complex c1;**

**cin >> c1;**

**cout << "The complex object is ";**

**cout << c1;**

**return 0;**

**}**

**3.** **#include <cstdlib>**

**#include <iostream>**

**using namespace std;**

**class Array {**

**private:**

**int\* ptr;**

**int size;**

**public:**

**Array(int\*, int);**

**int& operator[](int);**

**void print() const;**

**};**

**int& Array::operator[](int index)**

**{**

**if (index >= size) {**

**cout << "Array index out of bound, exiting";**

**exit(0);**

**}**

**return ptr[index];**

**}**

**Array::Array(int\* p = NULL, int s = 0)**

**{**

**size = s;**

**ptr = NULL;**

**if (s != 0) {**

**ptr = new int[s];**

**for (int i = 0; i < s; i++)**

**ptr[i] = p[i];**

**}**

**}**

**void Array::print() const**

**{**

**for (int i = 0; i < size; i++)**

**cout << ptr[i] << " ";**

**cout << endl;**

**}**

**int main()**

**{**

**int a[] = { 1, 2, 4, 5 };**

**Array arr1(a, 4);**

**arr1[2] = 6;**

**arr1.print();**

**arr1[8] = 6;**

**return 0;**

**}**

**4.** **#include<iostream>**

**#include<stdlib.h>**

**using namespace std;**

**class student**

**{**

**string name;**

**int age;**

**public:**

**student()**

**{**

**cout<< "Constructor is called\n" ;**

**}**

**student(string name, int age)**

**{**

**this->name = name;**

**this->age = age;**

**}**

**void display()**

**{**

**cout<< "Name:" << name << endl;**

**cout<< "Age:" << age << endl;**

**}**

**void \* operator new(size\_t size)**

**{**

**cout<< "Overloading new operator with size: " << size << endl;**

**void \* p = ::operator new(size);**

**return p;**

**}**

**void operator delete(void \* p)**

**{**

**cout<< "Overloading delete operator " << endl;**

**free(p);**

**}**

**};**

**int main()**

**{**

**student \* p = new student("Mohit", 24);**

**p->display();**

**delete p;**

**}**

**5.** **#include <iostream>**

**#include <cstdlib>**

**void\* operator new(size\_t sz)**

**{**

**void\* m = malloc(sz);**

**std::cout<<"User Defined :: Operator new"<<std::endl;**

**return m;**

**}**

**void operator delete(void\* m)**

**{**

**std::cout<<"User Defined :: Operator delete"<<std::endl;**

**free(m);**

**}**

**void\* operator new[](size\_t sz)**

**{**

**std::cout<<"User Defined :: Operator new []"<<std::endl;**

**void\* m = malloc(sz);**

**return m;**

**}**

**void operator delete[](void\* m)**

**{**

**std::cout<<"User Defined :: Operator delete[]"<<std::endl;**

**free(m);**

**}**

**class Dummy**

**{**

**public:**

**Dummy()**

**{**

**std::cout<<"Dummy :: Constructor"<<std::endl;**

**}**

**~Dummy()**

**{**

**std::cout<<"Dummy :: Destructor"<<std::endl;**

**}**

**};**

**int main()**

**{**

**int \* ptr = new int;**

**delete ptr;**

**Dummy \* dummyPtr = new Dummy;**

**delete dummyPtr;**

**int \* ptrArr = new int[5];**

**delete [] ptrArr;**

**return 0;**

**}**

**6.** **#include <iostream>**

**using namespace std;**

**class Complex {**

**private:**

**int real, img;**

**public:**

**Complex(int r, int i)**

**{**

**real = r;**

**img = i;**

**}**

**void operator=(const Complex& C)**

**{**

**real = C.real;**

**img = C.img;**

**}**

**void print() { cout << real << "+i" << img << endl; }**

**};**

**int main()**

**{**

**Complex C1(2, 3), C2(4, 6);**

**cout << "BEFORE OVERLOADING ASSIGNMENT OPERATOR"<< endl;**

**cout << "C1 complex number: ";**

**C1.print();**

**cout << "C2 complex number: ";**

**C2.print();**

**C1 = C2;**

**cout << "AFTER OVERLOADING ASSIGNMENT OPERATOR" << endl;**

**cout << "C1 complex number: ";**

**C1.print();**

**cout << "C2 complex number: ";**

**C2.print();**

**return 0;**

**}**

**7.** **#include<iostream>**

**using namespace std;**

**class Complex {**

**private:**

**int real, imag;**

**public:**

**Complex(int r = 0, int i = 0) {real = r; imag = i;}**

**Complex operator + (Complex const &obj) {**

**Complex res;**

**res.real = real + obj.real;**

**res.imag = imag + obj.imag;**

**return res;**

**}**

**void print() { cout << real << " + i" << imag << '\n'; }**

**};**

**int main()**

**{**

**Complex c1(10, 5), c2(2, 4);**

**Complex c3 = c1 + c2;**

**c3.print();**

**}**

**8.**