**Q1. Write a C++ Program to shown the concept of operator overloading also define a specific problem where we can use this concept for solve the specific problem using the overload function**

#include <iostream>

using namespace std;

class Cal {

public:

static int add(int a,int b){

return a + b;

}

static int add(int a, int b, int c)

{

return a + b + c;

}

};

int main(void) {

Cal C; // class object declaration.

cout<<C.add(10, 20)<<endl;

cout<<C.add(12, 20, 23);

return 0;

}



**Q2. Write a C++ Program to overload all the Logical Operators in a single program using the friend function based concept.**

#include <iostream>

class Cents

{

private:

int m\_cents {};

public:

Cents(int cents) { m\_cents = cents; }

// add Cents + Cents using a friend function

friend Cents operator+(const Cents &c1, const Cents &c2);

int getCents() const { return m\_cents; }

};

Cents operator+(const Cents &c1, const Cents &c2)

{

// use the Cents constructor and operator+(int, int)

// we can access m\_cents directly because this is a friend function

return Cents(c1.m\_cents + c2.m\_cents);

}

int main()

{

Cents cents1{ 6 };

Cents cents2{ 8 };

Cents centsSum{ cents1 + cents2 };

std::cout << "I have " << centsSum.getCents() << " cents.\n";

return 0;

}



**Q3. Write a C++ Program to overload the subscript operator [] by writing the member operator function operator definition. Also the above overloaded operator work on the vector class.**

// Overloading operators for Array class

#include <cstdlib>

#include <iostream>

using namespace std;

// A class to represent an integer array

class Array {

private:

int\* ptr;

int size;

public:

Array(int\*, int);

// Overloading [] operator to access elements in array style

int& operator[](int);

// Utility function to print contents

void print() const;

};

// Implementation of [] operator. This function must return a

// reference as array element can be put on left side

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

{

if (index >= size) {

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

exit(0);

}

return ptr[index];

}

// constructor for array class

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;

}

// Driver program to test above methods

int main()

{

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

Array arr1(a, 4);

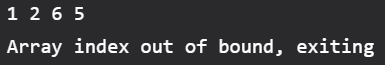
arr1[2] = 6;

arr1.print();

arr1[8] = 6;

return 0;

}



**Q5. Write a C++ program to overload the new and delete operator which is used for dynamically allocate and reallocated the memory for the creation and deletion of a object in a class respectively.**

#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);

//void \* p = malloc(size); will also work fine

return p;

}

void operator delete(void \* p)

{

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

free(p);

}

};

int main()

{

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

p->display();

delete p;

}

