**Session 4 (unit-2): Dynamic memory allocation and Inline vs. Macros**

1. **Show how can the Dynamic memory can be allocated and de-allocated to arrays in C++**

#include <iostream>

using namespace std;

int main()

{

int\*a; // declare an int pointer

float\* b; // declare a float pointer

a= new int; // dynamically allocate memory

b = new float;

\*a = 45; // assigning value to the memory

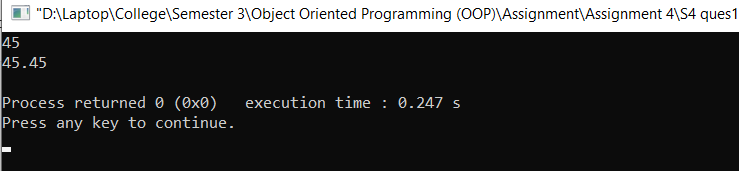
\*b = 45.45f;

cout << \*a<< endl;

cout << \*b<< endl;

delete a, b; // deallocate the memory

}



1. **Write a macro to calculate simple interest from principal, rate of interest and time. Simple interest = (principal\*rate of interest\*time)/100.**

#include<iostream>

using namespace std;

#define SI(p,r,t) (p\*r\*t)/100;

int main()

{

float p ;

float r ;

float t ;

cout<<"\nEnter Principal: ";

cin>>p;

cout<<"\n Enter rate: ";

cin>>r;

cout<<"\n Enter time: ";

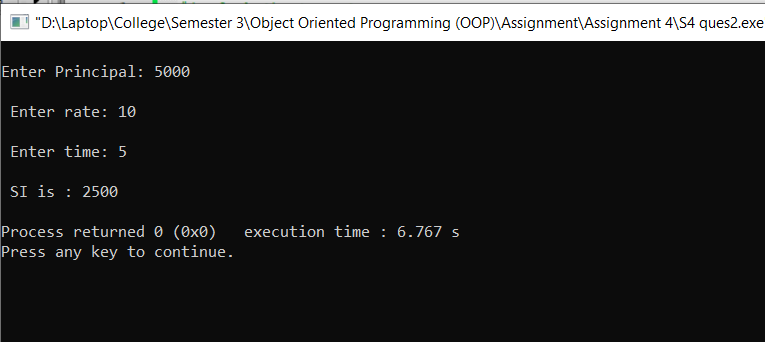
cin>>t;

float k = SI(p,r,t) ;

cout<<"\n SI is : "<<k<<endl;

return 0;

}



1. WAP that can demonstrate the functionality of a calculator and it must include the following operations:  
     
   addition, subtraction, division, multiplication and getting two input values from the user.

You have to perform implementation of all these operations using three cases:

Case 1: using normal functions

Case 2: using Inline functions

Case 3: using macros

Comment which will be the most suitable way of implementation in this particular problem and why?

**Case 1: using normal functions**

#include<iostream>

using namespace std;

int main()

{

char operator1;

int a,b;

cout<<"Enter an operator (+, -, \*,): ";

cin>>operator1;

cout<<"Enter two Numbers: ";

cin>>a>>b;

switch (operator1)

{

case '+':

cout<<a<<"+"<<b<<"="<<a + b;

break;

case '-':

cout<<a<<"-"<<b<<"="<<a - b;

break;

case '\*':

cout<<a<<"\*"<<b<<"="<<a \* b;

break;

case '/':

cout<<a<<"/"<<b<<"="<<a / b;

break;

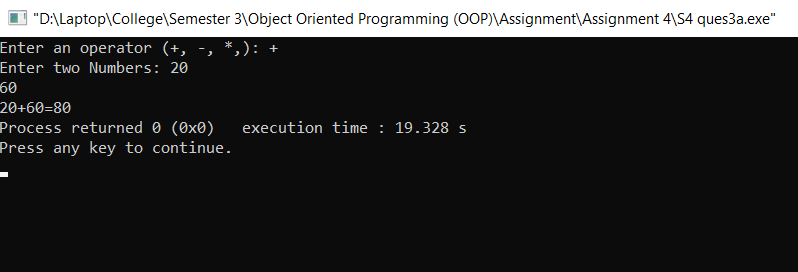
default:

cout<<"Error! operator is not correct";

}

return 0;

}



**Case 2: using Inline functions**

#include <iostream>

using namespace std;

class operators

{

int a,b,add,sub,mul;

float div;

public:

void getdata();

void addition();

void subtract();

void multiplication();

void division();

};

inline void operators :: getdata()

{

cout << "Enter first number:";

cin >> a;

cout << "Enter second number:";

cin >> b;

}

inline void operators :: addition()

{

add = a+b;

cout << "Addition of two numbers: " << a+b << "\n";

}

inline void operators :: subtract()

{

sub = a-b;

cout << "Subtraction of two numbers: " << a-b << "\n";

}

inline void operators :: multiplication()

{

mul = a\*b;

cout << "Multiplication of two numbers: " << a\*b << "\n";

}

inline void operators ::division()

{

div=a/b;

cout<<"Division of two numbers: "<<a/b<<"\n" ;

}

int main()

{

operators o;

o.getdata();

o.addition();

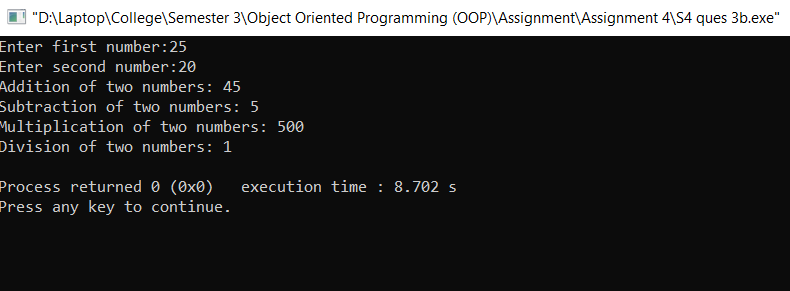
o.subtract();

o.multiplication();

o.division();

return 0;

}



**Case 3: using macros**

#include<iostream>

using namespace std;

#define ADD(a,b) (a+b)

#define SUBTRACT(a,b) (a-b)

#define MULTIPLICATION(a,b) (a\*b)

#define DIVISION(a,b) (a/b)

int main()

{

float a,b;

cout<<"Enter the number a :";

cin>>a;

cout<<"Enter the number b:";

cin>>b;

float add = ADD(a,b);

cout<<"Sum is :"<<add<<endl;

float sub = SUBTRACT(a,b);

cout<<"Difference is :"<<sub<<endl;

float mul = MULTIPLICATION(a,b);

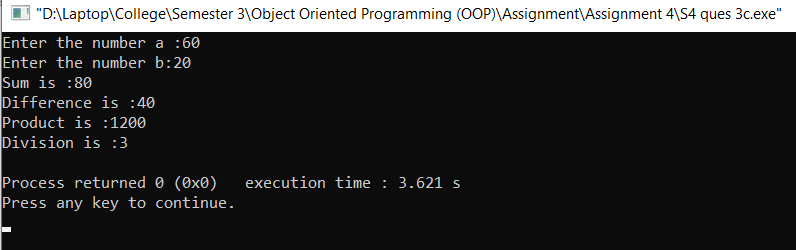
cout<<"Product is :"<<mul<<endl;

int div = DIVISION(a,b) ;

cout<<"Division is :"<<div<<endl;

return 0;

}



In the above example on Q-3 I believe that case 1 and case 3 example are most suitable way to implementation to do this practical.