KARTHIK V
CIT
HUAWEI INTERNSHIP
ASSIGNMENT

OOPS FOUNDATION

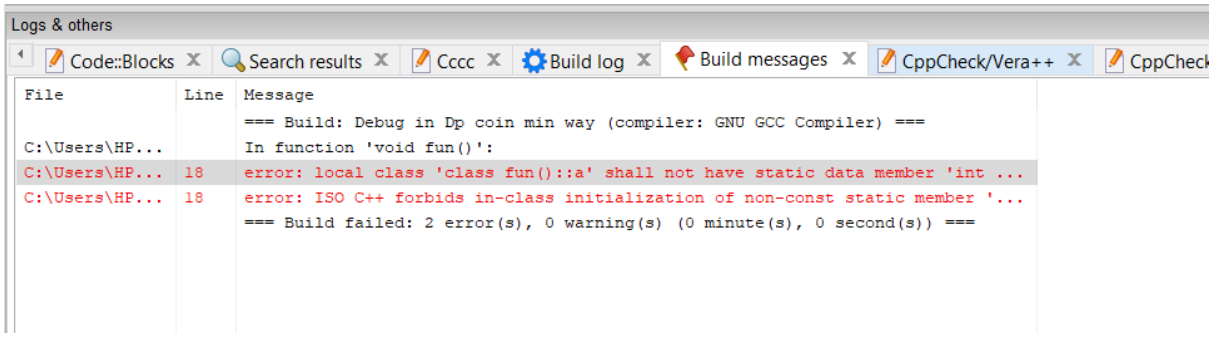
1. LOCAL CLASS CANNOT USE STATIC VARIBLES

local_class - Notepad

File Edit Format View Help

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

/*
Static variable cannot be used in local class because
1. static is initialized when program starts
   but local class defined when function is called
   and local class unloaded when function ends.
*/

void fun()
{   //local class of function fun()
        class a
        { public:

        //Invalid variable
        static int x=1;


        //valid variable
        int x=1;

        };
        a obj1;
        cout<<obj1.x;
}

int main()
{   fun();
        return 0;
}
```

OUTPUT


Logs & others
Code::Blocks
Search results
Cccc
Build log
Build messages
CppCheck/Vera++
CppCheck
File
Line
Message
=== Build: Debug in Dp coin min way (compiler: GNU GCC Compiler) ===
C:\Users\HP...
In function 'void fun()':
C:\Users\HP...
18
error: local class 'class fun()::a' shall not have static data member 'int ...
C:\Users\HP...
18
error: ISO C++ forbids in-class initialization of non-const static member '...
=== Build failed: 2 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===

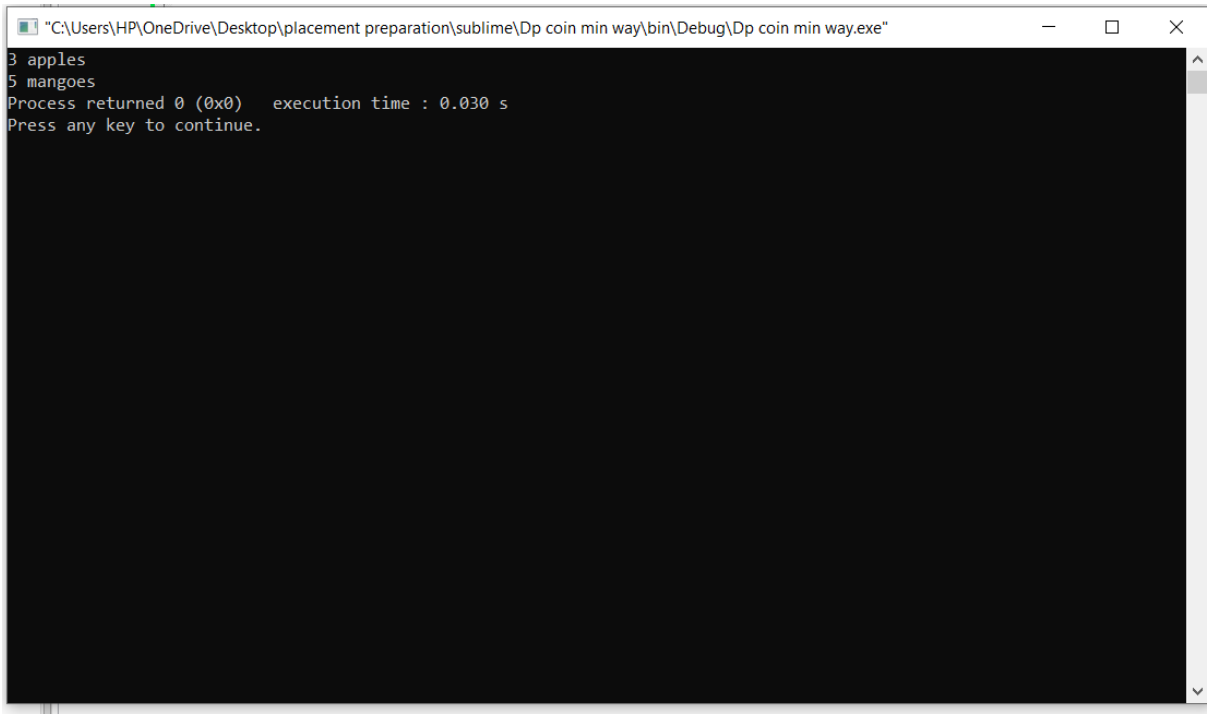
2. COPY CONSTRUCTOR

copy_constructor - Notepad

File Edit Format View Help

```
#include <iostream>

using namespace std;

class stock
{ public:
    int apple;
    int mango;

  //Two parameterized constructor for class stock
  stock(int apple,int mango)
  {
       //initialize class variables using integers as parameters
       this->apple=apple;
       this->mango=mango;
  }

  //Copy constructor for class stock
  stock(stock &b)
  {
       //initialize class variables using object a as parameter
       apple=b.apple;
       mango=b.mango;
  }
};
int main()
{
    //object a is created and parameterized constructor is called with two integer values
    stock a(3,5);
    //Object b is created and copy constructor is called with object a
    stock b(a);

    cout<<b.apple<<" apples\n"<<b.mango<<" mangoes";
    return 0;
}
```

OUTPUT


"C:\Users\HP\OneDrive\Desktop\placement preparation\sublime\Dp coin min way\bin\Debug\Dp coin min way.exe"
3 apples
5 mangoes
Process returned 0 (0x0)   execution time : 0.030 s
Press any key to continue.

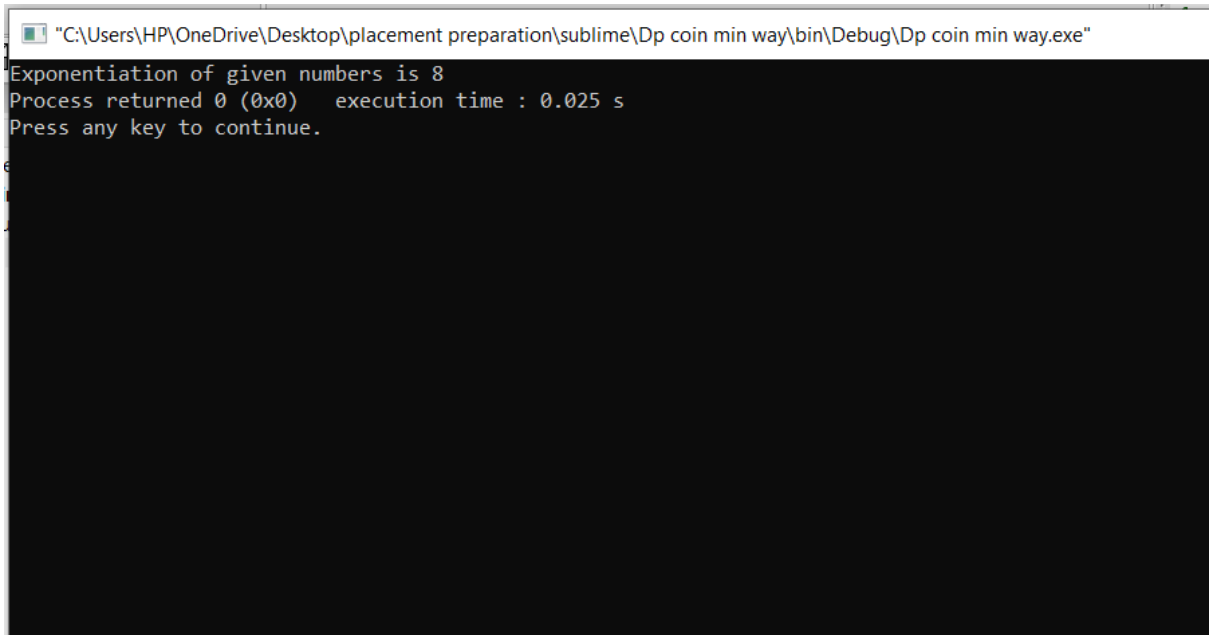
3. OPERATOR OVERLOADING

operator_overloading - Notepad

File Edit Format View Help

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

class maths
{ public:
    int num;
    maths(){}
    //parameterized constructor
    maths(int val)
    { num=val;

    }
    //operator overloading function
    maths operator^(maths b)
    {  //calculate power of object a(num) and 'b.num'
       //then store ans in 'm' and return object 'm'
       maths m;
       m.num=pow(num,b.num);
       return m;

    }
    void print()
    {
       cout<<"Exponentiation of given numbers is "<<num;
    }
};
int main()
{  //Two objects are created and called parameterized constructor
   maths a(2),b(3);
   //Using ^ , operator overloading function is called
   //to do the powers of object 'a' value and object 'b' value
   maths c=a^b;
   //call print function using object 'c'
   c.print();
   return 0;
}
```

OUTPUT


"C:\Users\HP\OneDrive\Desktop\placement preparation\sublime\Dp coin min way\bin\Debug\Dp coin min way.exe"
Exponentiation of given numbers is 8
Process returned 0 (0x0)   execution time : 0.025 s
Press any key to continue.

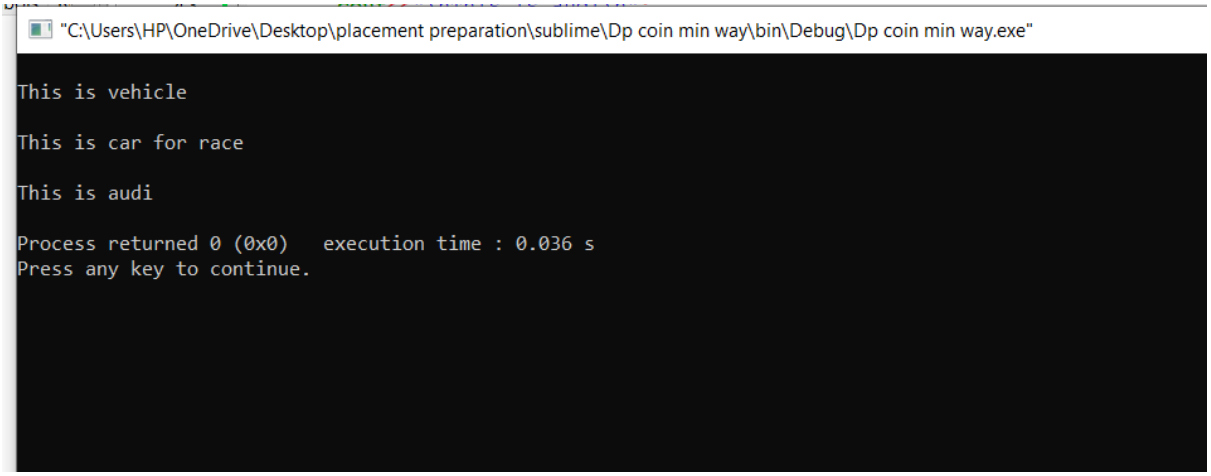
4. HYBRID INHERITANCE

hybrid_inheritance - Notepad

File Edit Format View Help

```
#include <iostream>

using namespace std;

//single inheritance
class vehicle
{
public:
    vehicle()
    {
        cout<<"\nThis is vehicle\n";

    }
};

//Car derived from vehicle
class car:public vehicle
{
public:
    car()
    {
        cout<<"\nThis is car";

    }
};

class race
{

public:
    race(){
     cout<<" for race\n";
    }
};

//multiple inheritance
//Audi derived from car and race
class audi:public car,public race
{
public:
    audi()
    {
       cout<<"\nThis is audi\n";
    }
};

int main()
{
    //Object is created for class audi
    audi mine;
    return 0;
}
```

OUTPUT


"C:\Users\HP\OneDrive\Desktop\placement preparation\sublime\Dp coin min way\bin\Debug\Dp coin min way.exe"
This is vehicle
This is car for race
This is audi
Process returned 0 (0x0)   execution time : 0.036 s
Press any key to continue.

5. STACK AND QUEUE

templates - Notepad

File Edit Format View Help

```
#include <iostream>
//Header files for stack and queue
#include <stack>
#include <queue>

using namespace std;

void stack_operation()
{
    //Creation of stack and it uses LIFO order
    stack<int> stk;

    //Insertion of elements in stack using push function one by one
    stk.push(1);
    stk.push(2);
    stk.push(3);

    //Size of created stack
    cout<<"\nThe stack size is: "<<stk.size();


    //top function is used to print lastly inserted element in stack
    cout<<"\nThe top element of stack is: "<<stk.top();

    //empty function is used to check if stack is empty or not
    //pop function is used to delete lastly inserted element in stack
    cout<<"\nThe elements in stack are:";
    while (!stk.empty())
    {
                cout << ' ' << stk.top();
                stk.pop();
    }
}

void queue_operation()
{
    //Creation of queue and it uses FIFO order
    queue<int> que;

    //Insertion of elements in queue using push function one by one
    que.push(1);
    que.push(2);
    que.push(3);
```

```
    //Size of created queue
    cout <<"\n\nThe queue size is: "<< que.size();

    //back() is used to get last element in queue
    cout <<"\nThe last element in queue is: "<< que.back();

    //front function is used to print first element in queue
    cout <<"\nThe first element in queue is: "<< que.front();

    //empty function is used to check if queue is empty or not
    //pop function is used to delete the first element in queue
    cout<<"\nThe elements in queue are:";
    while (!que.empty())
    {
        cout << ' ' << que.front();
        que.pop();
    }

}


int main()
{   //Calling respective stack and queue functions
    stack_operation();
    queue_operation();

    return 0;

}
```

OUTPUT