Date: 13/01/20

* Properties of OOP:

1) Data Abstraction (Hiding the background details)
2) Data hiding
3) Data Encapsulation
4) Inheritance
5) Polymorphism.

(8) Write a program in c++ using class to input and output the data.

Solve:

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

class Display
{ int num1 , num2;
  public: void input();
          void output();
};

void Display::input(){
    cout<<"Input two numbers:";
    cin>>num1>>num2;
}

void Display::output(){
    cout<<num1<<num2;
}

int main()
{ Display ob;
  ob.input();
  ob.output();
  return 0;
}
```

Q) Write a class Calculator. (from user).

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

Class calculator
{ int num1, num2;
  Public: void add();
          void sub();
          void mul();
          void Div();
          void input();
void add() {
    cout << num1+num2;
    }

Void sub()
   { cout << num1 - num2; }

void mul()
   { cout << num1* num2; }

void Div()
   { cout << num1 / num2; }

void input()
   { cout << "Enter two number:";
     cin >> num1 >> num2;
   }
};
```

```
int main()
{
    Calculator ca;

    ca.input();
    ca.add();
    ca.sub();
return 0;
}
```

output: → Enter two number : 10 5
→ 15
→ 5

Q.)
```
#include <iostream>
using namespace std;

int num1, num2;

int area(int a, int b)
{ num1 = a;
  num2 = b;

  int c = a*b;

  return c;
}
```

```
int main ()
{  area (10, 5) ;  }
```

Or.

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

Class area{
        int a, b ;
        cout << "Enter two number to find area";
        cin >> a >> b;

Public : void Area()
        { cout << "Enter two number to bind area";
          cin >> a >> b;
          Cout<< " Area of room is : " << a*b ;
} };

int main()
    { area AR();
        Ar. Area() ;
    }
```

(Q) Write a program to create an Inventory management system and create a Bill on receipt.

Product price Quanty

Amount GST

Shortcut Linux.

Pwd → Present working directory (showing)
ls → list (showing)
mkdir Name → box boldor making

Date → 20/11/

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

Class Card
    { Char title[50];
      Char name[40];
      Int number;
Public: void store (Char *t , Char * author ,
        void show ();
    };

Void Card:: store (Char *t , Char * author,
    { strcpy (title , t);
      strcpy ( name , author);
        number = num;
    }
Void Card:: show()
    { cout << "title:" << title;
      cout << "author:" << name;
      cout << number;
    }

Int main()
  { Card book1;
```

gedit file.c/cpp (to create a c or cpp file)

```
book1.store("oop","BSwamy",3);
book1.show();
}
```

Q) Explain Properties of oop Explain with example.
Q) How many acces specifier. What is its role member function, member data. Scope of class, when.

Q) What is data hiding why important.
Q) How many type of polymorphism.
Q) What is scope resolution operator.

* Function overloading :-

```
int func1(int x);    float func1(int x
```

```
int func1(int x)
{
    a=x;
    return a;
}

float func1(float y)
{
    b=y;
    return b;
}
```

* write a program to overload add() to 2 int and 2 bloat using class

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

int add (int a, int b)
{
    int c = a+b;
    return c;
}

int add (bloat a, bloat b)
{
    bloat c = a+b;
    return c;
}

int main()
{
    int a=10;
    int b=5;
    int c= add(a,b);
    cout << c << endl;

    float d = 5.4;
    bloat e = 4.6;
    bloat f = add(d,e);
    cout << f << endl;
}
```

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

class method {
    public: int add(int x, int y);
            float add(float x, float y);
};

int method :: add(int x, int y)
    { return x+y; }

float method :: add(float x, float y)
    { return x+y; }

int main()
{
    method add1;
int x=
int y=
    add1.add(10, 5);
    add1.add(4.6, 5.4);
    cout<< x << y;
}
```

* → **Constructor function** is a function having the same name as that of the class. It doesn't having return type. It can take any no. of parameter. It is automatically called when a object is created. The purpose of this function is to initialise the member data.

Ex→ Class samp
{ int a;
Public : samp()
{ a = 0;
Cout << "constructing" << endl;
} };

* → **Destructor function** is the function with the same name as class with any return type and tilde (~) character preceding the function name. It doesn't take any parameter. It is called when a object is goes out of scope. It's purpose is to de allocate the memory and object.

Ex→ Class samp
{ int a;
Public : samp()
{ a=0;
cout << a;
}
~ samp() { cout << "destructing" }

int main()
{ { samp ob; }
return 0;
}

Q. Why constructor is overload and why not destructor is overload.

(A).

```
Class Samp
{ Public : int a;
    void samp()
      {a=0;}
};
Int main()
{ Samp ob; }
```

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

Class times
{ Clock_t start, end;
  Public : timer();
           ~timer();
};

timer : timer()
{ start = clock(); }

timer : : ~timer()
{ end = clock();
  Cout << " time" << end-start;
}
```

```
int main()
{ { timer ob;
    Char ch;
  cout << "Enter a ch" << endl;
    Cin >> ch; }
  return 0;
}
```

(*) Create a class stopwatch initially set elapsed time to 0 declare two mem start and end. Start will turn 'on' the stopwatch. end will turn "off". Use a class member show to show the elapsed time and a destructor function to show time.

Solve→

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

Class stopwatch
{ clock_t start, end;
  ~~public: void show();~~ begin = end = 0;
    ~~timer();~~ clock_t elapsedtime
public: stopwatch()
        {
        void start()
          {begin = clock(); }
        void stop()
          { end = clock(); }
```

```
void show()
    { cout<< end-begin; }

~ stopwatch()
    { { show(); } };

int main()
{   stopwatch ob;
    ob.start();
for (int i=0; i<100; i++)
    {   ob.end(); }
    }
```

Date 24/01/2023

Q. Write a CPP program to create a data store for books. The data member should be book name, author, ISBN, date of Publishing. accept the data in a well formated manner. & Print

Q. (2) create a class mobile phone. Accept data for n number and displayed the same in well formated manner.

(3) Create a class event Excepts the event detail and print the event tickets for the participation. Also enter some predefined data like College name, branch name and date.

④ Write a programm with neccessary data member to prove that a default constructor get executed even though no explict constructor is declared.

⑤ Write a programm to Enter 10 words and string of 3 long sentences, implement a function to calculate no. of words present in a given strings and remaing words use a member function to displayed matched word with frequency of occurences, remaining words in the string.

⑥ Using Array of object concept implement the above programm for multiple strings.

⑦ ~~Write a programm~~

Date 02/02/23

```
* Class samp
    { int a,b;
      public: samp()
        { cout<<"default";}
        samp(int x, int y)
        { a=x;
          b=y;}
      ~ samp()
        { cout<<a<<b;}
```

```
int main()
  { { Samp ob;
      Samp ob(10, 20);
    }
    return 0;
  }
```

→

(3). Create a class called Prompt. Pass its constructor a prompting string and display the string. Let the user input a number and store the number in a private member count. Ring the bell on the terminal when the object goes out of scope as many times as user has entered.

Ans→

```
Class prompt
{ Public: prompt (string s)
          { this->s = s;
            cout << s << endl; }
  int a;
  cin >> a;
  Private: int count(int n)
           { a = n; }
```

Ans.

```
Class prompt{
    int count;
public : Prompt.(string s)
    { cout << s;
      cin >> count;
    }
~ Prompt()
    { for(int i=0 ; i< count; i++)
      { cout << "\n"; }
    }

int main()
{ { prompt ob("enter a number");
  }
  return 0;
}
```

Q). Create a class called line. Let the constructor take the length of the line as parameter. and store it in private member length. Let the destructor draw a line on the terminal. as long as you have entered.

Ans> Class line{

~~int length;~~

public : line(~~int length~~)

{ cout << "Enter the length";

cin >> length;

}

```
class line { int length;
public: line(int length)
{ this->length = length; }

~ line ()
{ for (int i=0 ; i< length; i++)
{ cout << "------" << endl; }
};

int main()
{ { line ob(10); }
return 0;
}
```

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

int static x=0;

class samp
{ public: samp()
{ cout << ++x; }

~ samp()
{ cout << "destructing" << x;
} };

int main();
{ samp ob1, ob2, ob3; }
```

Example of constructor are called in order and destructor in reverse order

```
class dyna
{ public: dyna()
  { cout<<" Hello";
  }
 ~dyna()
  { cout <<" out"; }
};

int main()
 { { dyna ob1; }
   dyna ob2, ob3;
 }
```

Output → Hello out Hello Hello out out.

* Passing objects to function

```
Class samp
   { int i;
   Public: samp(int n) {i=n;}
     int get_i() {return i;}
   };

int sqr_it (samp o)
   { return o.get_i() * o.get_i();
   }
int main()
   { samp a(10), b(2);
     cout << sqr_it (a);
     cout << sqr_it (b);
```

8)

```
for(int i=0 ; i<3; i++)
{static char ch = 'A';
 cout << ch++;
}
```

* inline function → When a function is written in a single line it is called inline function or when function name preceds the inline keyword then also it is inline function.

Advantage :

(1) Take less time to compile.
(2) function call is replaced by function code.
(3) function should be written first.
(4) No loop is allowed.

Example →

```
#inline void b()
{ cout<<"Hello";
}
void b(){ cout<<"Hello";}
int main()
{ b();
}
```

9) write a class mydetails. having constructor function take the parameter age, class Rollno. and store in member function include a display function. to show the

```
Class my_details {
    int age = a, class = b, roll = c;
Public: my_details (int age, int class, int roll)
    {
        a = age;
        b = class;
        c = roll;
    }
    void show();
    ~my_details()
    { cout<< a << " = age" << endl
      cout<< b << " = class" << endl
      cout<< c << " = roll" << endl;
    } };

void my_details :: show()
    { cout<< a << " = age" << endl;
      cout<< b << endl;
      cout<< c << endl;
    }

int main() {
    my_details S1(20, 5, 007);
    S1.show();
    return 0;
}
```

Date → 3/2/23

* object call by value.

```
class samp
{ int i;
  public: samp(int n)
          { i=n; }
  void set_i(int n){ i=n; }
  int get_i(){return i;}
};

void sqr_it(samp o)
{ o.set_i(o.get_i() + o.get_i());
cout<<"copy ob a has i value ob"<<o.get_i();
}

int main()
{   samp a(10);
    sqr_it(a);
  cout<<"But a is unchanged in main.";
  cout<<a.get_i();
    return 0;
}
```

* object call by reference.

Only change void sqr_it(samp *o)
and sqr_it(&a);
in upper code

* Find the o/p.

```
Class samp.
  {int i;
   Public : samp(int n)
     { i = n;
     cout << " constructing" ;
     }
  ~ samp() { cout << " Destructing" ; }
     int get_i() {return i ;}
  };
    int sqr_it (samp o)
    {return o.get_i() * o.get_i();}

int main()
  { samp a(10);
    Cout << sqr_it (a);
    return 0;
  }
```

Output: Constructing Destructing 100
Destructing

Q. Create a class who have Whose Constructor ~~called~~ take one character argument that will will used to identify a object . have the constructor Display a message " constructing who #

When a object is destroy have a message destroy who the. Finally create a function who. that returns who object. Give each object a unique name.

Solve→

```
Class who
  { Char ch;
    public: who (Char ch)
      { c = ch;
        for(int i=0; i< object.length); i++)
        { if (A[i] == 'c')
          {cout << "constructing who" << c;}
        }
      }
    ~ who

Class who.
  { Char name;
    Public: who (Char c)
        { name = c;
      cout << " constructing who # " << name;
    }
  ~ who()
    { cout << " Destroying who # " << name.
  }  };

Who make_who()
    { who temp ('B');
      return temp; }.
int main()
    { who Ob ('A');
      make_who();
```

* What is friend function:- friend function is a non member function which is declare inside a class with a "friend" keyword with taking an object as parameter. though it is non member function it can access private and protected of class. The definition is written outside the class. The syntax of function is " friend void s(object)".