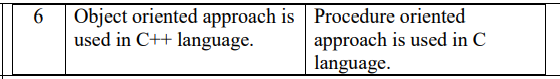
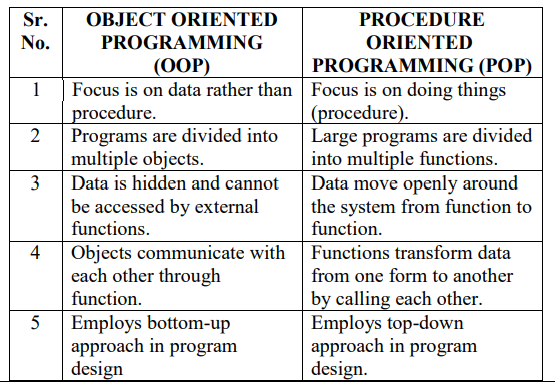
**QUESTION BANK [OOP]**

**. State any 4 OOP Languages. W-18 2M**

* **C++**
* **Smalltalk**
* **Object pascal**
* **java**
* **Simula**
* **Ada**
* **Turbo pascal**
* **Eiffel**
* **C#**
* **Python**

**. Difference between OOP and POP (W-18) (W-19) (2M)**



**. Write a C++ program to accept array of 5 elements, find and display smallest number from an Array. (W-18)(4M)**

#include<iostream.h>

#include<conio.h>

void main()

{

int a[5],smallest,i;

clrscr();

cout<<" Enter array elements:";

for(i=0;i<5;i++)

cin>>a[i];

smallest=a[0];

for(i=1;i<5;i++){

if(a[i]<smallest){

smallest=a[i];

}

}

cout<<endl<<"Smallest number="<<smallest;

getch();

}

**. Write a C++ program to print multiplication table of 7. (W-18)(4M)**

#include<iostream.h>

#include<conio.h>

void main()

{

int num;

clrscr();

cout<<"Multiplication table for 7 is:"<<endl;

for(num=1;num<=10;num++)

{

cout<<"7 \*"<<num<<"="<<7\*num<<endl;

}

getch();

}

**. a) C++ program to find even or odd.(W-18) (6M)**

#include<iostream.h>

#include<conio.h>

void main()

{

int num;

clrscr();

cout<<"\nEnter a Number ";

cin>>num;

if(num%2==0)

{

cout<<"\nEntered number is even";

}

else

{

cout<<"\nEntered number is odd";

}

getch();

}

**b) C++ program to declare structure employee with members as empid and empname. Accept and display data for one employee using structure variable.**

#include<iostream.h>

#include<conio.h>

struct employee

{

int empid;

char empname[10];

};

void main()

{

employee e;

clrscr();

cout<<"\nEnter employee id and Employee Name ";

cin>>e.empid>>e.empname;

cout<<"\mThe Employee Id is "<<e.empid;

cout<<"\nThe Employee Name is "<<e.empname;

getch();

}

**. State use of cin and cout. (S-19) (2M)**

cin: cin is used to accept input data from user (Keyboard). cout:cout is used to display output data on screen.

**. State use of scope resolution operator (S-19) (W-19) (2M)**

It is used to uncover a hidden variable. Scope resolution operatorallows access to the global version of a variable. The scope resolution operator is used to refer variable of class anywhere in program.

:: Variable name.

OR

Scope resolution operator is also used in classes to identify the class to which a member function belongs. Scope resolution operator is used to define function outside of class. Return\_type class\_name:: function\_name( ) { Function body }

**. Write a C++ program to find factorial of a number. (S-19) (4M)**

#include<iostream.h>

#include<conio.h>

void main()

{

int no,fact=1,i;

clrscr();

cout<<"Enter number:";

cin>>no;

for(i=1;i<=no;i++)

{

fact=fact\*i;

}

cout<<"Factorial ="<<fact;

getch();

}

**1O. Accept data for 10 students and display it. Write a C++ program to display sum of array elements of array size n (S-19) (4M)**

#include<iostream.h>

#include<conio.h>

void main()

{

int arr[20],i,n,sum=0;

clrscr();

cout<<"\nEnter size of an array:";

cin>>n;

cout<<"\nEnter the elements of an array:";

for(i=0;i<n;i++)

{

cin>>arr[i];

}

for(i=0;i<n;i++)

{

sum=sum+arr[i];

}

cout<<"\nArray elements are:";

for(i=0;i<n;i++)

{

cout<<arr[i]<<" ";

}

cout<<"\nSum of array elements is:"<<sum;

getch();

}

**1. Describe following terms: Inheritance, data abstraction, data encapsulation, dynamic binding (S-9) (4M)**

Inheritance:

1. Inheritance is the process by which objects of one class acquire the properties of objects of another class.

2. It supports the concept of hierarchical classification. It also provides the idea of reusability.

Data abstraction:

1. Data abstraction refers to the act of representing essential features without including the background details or explanations.

2. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and functions to operate on these attributes.

Data encapsulation:

1. The wrapping up of data and functions together into a single unit (called class) is known as encapsulation.

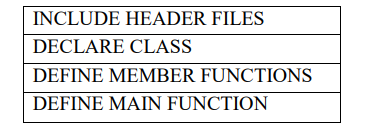
2. By this attribute the data is not accessible to the outside world, and only those functions which are wrapped in the class can access it.

Dynamic Binding:

1. Dynamic binding refers to the linking of a procedure call to be executed in response to the call.

2. It is also known as late binding. It means that the code associated with a given procedure call is not known until the time of the call at run-time.

**2. a) Describe structure of C++ program**



Description:-

1. Include header files

In this section a programmer include all header files which are require to execute given program. The most important file is iostream.h header file. This file defines most of the C++statements like cout and cin. Without this file one cannot load C++ program.

2. Declare Class

In this section a programmer declares all classes which are necessary for given program. The programmer uses general syntax of creating class.

3. Define Member Functions

This section allows programmer to design member functions of a class. The programmer can have inside declaration of a function or outside declaration of a function.

4. Define Main Functions

This section the programmer creates object and call various functions writer within various class.

**b) C++ program to add two 3\* 3 matrix and display addition**

#include<iostream.h>

#include<conio.h>

void main()

{

clrscr();

int mat1[3][3], mat2[3][3], i, j, mat3[3][3];

cout<<"Enter matrix 1 elements :";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

cin>>mat1[i][j];

}

}

cout<<"Enter matrix 2 elements :";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

cin>>mat2[i][j];

}

}

cout<<"Adding the two matrix to form the third matrix\n";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

mat3[i][j]=mat1[i][j]+mat2[i][j];

}

}

cout<<"The two matrix added successfully...!!";

cout<<"The new matrix will be :\n";

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

cout<<mat3[i][j]<<" ";

}

cout<<"\n";

}

getch();

}

**13. Write 4 benefits of OOP (W-19) (4M)**

Benefits Of OOP:

l. We can eliminate redundant code and extend the use of existing classes.

2. We can build programs from the standard working modules that communicate with one another, rather than having to start writing the code from scratch. This leads to saving of development time and higher productivity.

3. The principle of data hiding helps the programmer to build secure

programs that cannot be invaded by code in other parts of the

program.

4. It is possible to have multiple instances of an object to co-exist without any interference.

5. It is possible to map objects in the problem domain to those in the program.

6.It is easy to partition the work in a project based on objects.

7.The data-centered design approach enables us to capture more

details of a model in implementable form.

8.Object-oriented systems can be easily upgraded from small to

large systems.

9.Message passing techniques for communication between objects

makes the interface descriptions with external systems much

simpler.

10. 10. Software complexity can be easily managed.

**14. Write applications of OOP (W-19) (4M)**

Applications Of Object oriented programming are:

1)Real time systems

2)Simulation and modeling

3)Object-oriented databases

4)Hypertext, hypermedia and expertext

5)A1 and expert systems

6)Neural networks and parallel programming

7)Decision support and office automation systems

8)CIM/CAM/CAD systems

**15. Write a program to sort I-d array in ascending order (W-19) (4M)**

#include<iostream.h>

#include<conio.h>

void main()

{

int arr[20];

int i, j, temp,n;

clrscr();

cout<<"\n Enter the array size:";

cin>>n;

cout<<"\n Enter array elements:";

for(i=0;i<n;i++)

{

cin>>arr[i];

}

for(i=0;i<n;i++)

{

for(j=i+1;j<n;j++)

{

if(arr[i]>arr[j])

{

temp=arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

}

cout<<”Sorted Array:”;

for(i=0;i<n;i++)

{

cout<<”\n”<<arr[i];

}

getch();

}

**16. Difference between: Class & Structure**

|  |  |
| --- | --- |
| class | structure |
| Classes can contain constructor or destructor. | Structure does not contain parameter less constructor or destructor |
| It supports inheritance. | It does not support inheritance. |
| It binds data as well as function. | It can only handle data. |
| Function member of the class can be virtual or abstract. | Function member of the struct cannot be virtual or abstract. |
| Class members are private by default. | Structure member are public by default. |
| It is reference type | It is value type. |

**17. Program to check Armstrong number or not**

#include<iostream.h>

#include<conio.h>

int main(){

clrscr();

int n,rem,result=0,og;

cout<<"\n\n\nSHAIKH AZLAN AHMED\t210454\n";

cout<<"pls.enter a three digit number:";

cin>>n;

og=n;

while(n>0){

rem=n%10;

result+=rem\*rem\*rem;

n=n/10;

}

if(result==og)

cout<<"Armstrong Number";

else

cout<<"Not an armstrong number";

getch();

return 0;

}

**18. Difference between: C and C++**

|  |  |
| --- | --- |
| c | C++ |
|  |  |
| 1.C is a middle level language. | 1.C++ is a high level language. |
| 2.Top down approach is used in program design. | 2.Bottom up approach adopted in program design. |
| 3.printf() and scanf() is used for output and input. | 3.cin>> and cout<< is used for input and output. |
| 4.data cannot be hidden. | 4.data can be hidden. |

**19. Program to check whether a number is palindrome or not**

#include<iostream.h>

#include<conio.h>

int main(){

int rem,rev=0,op,n;

clrscr();

cout<<"\n\n\nSHAIKH AZLAN AHMED\t210454\n";

cout<<"\nEnter digit:";

cin>>n;

op=n;

while(n>0){

rem=n%10;

n=n/10;

rev=rem+rev\*10;

}

if(op==rev)

cout<<op<<" is palindrome";

getch();

return 0;

}

**20. Program to print a matrix: 1**

**23**

**456**

#include<iostream.h>

#include<conio.h>

int main(){

clrscr();

int m,n,i=1,r,c;

cout<<"\n\n\nSHAIKH AZLAN AHMED\t210454\n";

cout<<"Enter rows and columns:";

cin>>m>>n;

for(r=1;r<=m;r++){

for(c=1;c<=r;c++){

cout<<i<<"\t";

i++;

}

cout<<"\n";

}

getch();

return 0;

}

**21. Program to print a \* pyramid**

#include <iostream>

using namespace std;

int main()

{

int space, rows;

cout<<"SHAIKH AZLAN AHMED\t210454";

cout <<"\nEnter number of rows: ";

cin >> rows;

for(int i = 1, k = 0; i <= rows; ++i, k = 0)

{

for(space = 1; space <= rows-i; ++space)

{

cout <<" ";

}

while(k != 2\*i-1)

{

cout << "\* ";

++k;

}

cout << endl;

}

return 0;}

**CHP 2:**

**1. Describe use of protected access specifier (W-18) (2M)**

Protected access specifier is use to declare a class member that is

accessible by the member functions within its class and any class

immediately derived from it.

**2. Write any 2 characteristics of destructor (W-18) (2M)**

Characteristics:

1. It is used to destroy objects created by a constructor.

2. Name of destructor and name of the class is same.

3. Its name is preceded with tilde (~) symbol.

4. It never takes any argument.

5. It does not return any value.

6. It is invoked implicitly by the compiler upon exit from the

program (or block or function) i.e when scope of object is over.

**3. Write a C++ program to declare class addition with data members as x and y with constructor.Calculate addition & display it using function display. (W-18) (4M)**

#include<iostream.h>

#include<conio.h>

class addition

{

int x,y;

public:

addition(int,int);

void display();

};

addition::addition (int x1,int y1)

{

x=x1;

y=y1;

}

void addition::display()

{

cout<<"\nAddition of two numbers is:"<<(x+y);

}

void main()

{

addition a(3,4);

a.display();

getch();}

**4. Describe use of static data member in C++ with example (W-18) (4M)**

Use of static data member:

1. Static data member is used to maintain values common to the entire

class.

2. It is initialized to zero when the first object of its class is created.

3. Only one copy of that member is created for the entire class and is

shared by all the objects of that class.

Example:

#include<iostream.h>

#include<conio.h>

class test

{

static int count;

int obj\_no;

public:

void getdata(){

obj\_no=++count;

cout<<"\n Object number="<<obj\_no;

}

static void showcount()

{

cout<<"\n total number of objects="<<count;

}

};

int test::count;

void main()

{

test t1,t2;

clrscr();

t1.getdata();

t2.getdata();

test::showcount();

test t3;

t3.getdata();

test::showcount();

getch();

}

**5. Write a C++ program to find greatest number among two numbers from two different classes using friend function. (W-18) (4M)**

#include<iostream.h>

#include<conio.h>

class second;

class first

{

int x;

public:

void getx()

{

cout<<"\nEnter the value of x:";

cin>>x;

}

friend void max(first,second);

};

class second

{

int y;

public:

void gety()

{

cout<<"\nEnter the value of y:";

cin>>y;

}

friend void max(first,second);

};

void max(first a,second b)

{

if(a.x>b.y)

{

cout<<"\Greater value is:"<<a.x;

}

else

{

cout<<"\nGreater value is:"<<b.y;

}

}

void main()

{

first a;

second b;

clrscr();

a.getx();

b.gety();

max(a,b);

getch();

}

**6. Write a C++ program to declare class 'account' with data members as acc\_no, name and balance.Accept data for 8 accounts and display details of accounts having balance less than 10000. (W-18)(6M)**

#include<iostream.h>

#include<conio.h>

class Account

{

long int accno, bal;

char name[10];

public:

void getdata()

{

cout<<"\nEnter account number, balance and name ";

cin>>accno>>bal>>name;

}

void putdata()

{

if(bal>10000)

{

cout<<"\nThe Account Number is "<<accno;

cout<<"\nThe Balance is "<<bal;

cout<<"\nThe Name is "<<name;

}

}

};

void main()

{

Account a[8];

int i;

clrscr();

for(i=0;i<8;i++)

{

a[i].getdata();

}

for(i=0;i<8;i++){

a[i].putdata();

}

getch();

}

**4. Define class and object (S-19) (2M)**

Class: Class is a user defined data type that combines data and functions together. It is a collection of objects of similar type.

Object: It is a basic run time entity that represents a person, place or any item that the program has to handle.

**5. Describe use of static data member (S-19) (2M)**

Use of static data member:

Static data member (variable) is used to maintain values common to the entire class. Only one copy of static member is created for the entire class and is shared by all the objects of that class. Its lifetime is the entire program.

**6. Write C++ program to find smallest number from 2 nos. using friend function (S-19) (4M)**

#include<iostream.h>

#include<conio.h>

class class2;

class class1

{

int no1;

public:

void get1()

{

cout<<"Enter number 1:";

cin>>no1;}

friend void smallest(class1 no1,class2 no2);

};

class class2{

int no2;

public:

void get2(){

cout<<"Enter number 2:";

cin>>no2;

}

friend void smallest(class1 no1,class2 no2);

};

void smallest(class1 c1,class2 c2){

if(c1.no1<c2.no2)

cout<<"no1 is smallest";

else

cout<<"no2 is smallest";

}

void main(){

class1 c1;

class2 c2;

clrscr();

c1.get1();

c2.get2();

smallest(c1,c2);

getch();

}

**7. Write a C++ program to create class STUDENT. The data members of student class: Roll\_no, name,marks (S-19) (4M)**

#include<iostream.h>

#include<conio.h>

class STUDENT{

int Roll\_No;

char Name[20];

float Marks;

public:

void Accept();

void Display();

};

void STUDENT::Accept(){

cout<<"\nEnter data of student:";

cout<<"\nRoll number:";

cin>>Roll\_No;

cout<<"\nName:";

cin>>Name;

cout<<"\nMarks:";

cin>>Marks;

}

void STUDENT::Display(){

cout<<"\nStudents data is:";

cout<<"\nRoll number:"<<Roll\_No;

cout<<"\nName:"<<Name;

cout<<"\nMarks:"<<Marks;

}

void main(){

STUDENT S[5];

int i;

clrscr();

for(i=0;i<5;i++){

S[i].Accept();

}

for(i=0;i<5;i++){

S[i].Display();

}

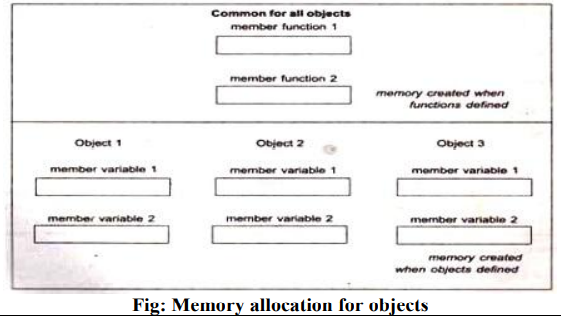
getch();

}

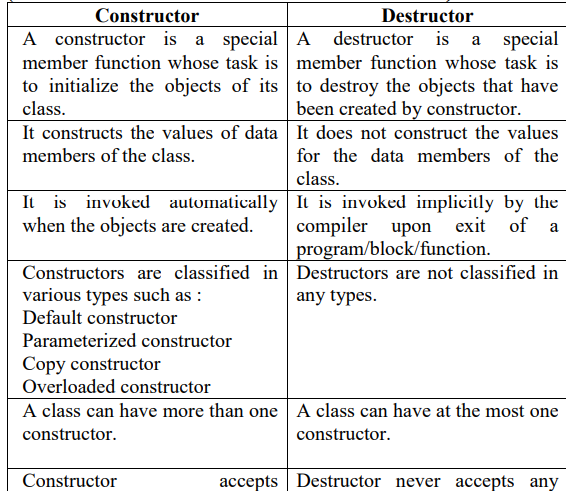
**8. Describe how memory is allocated to objects of class with suitable diagram (S-19) (W-19) (4M)**

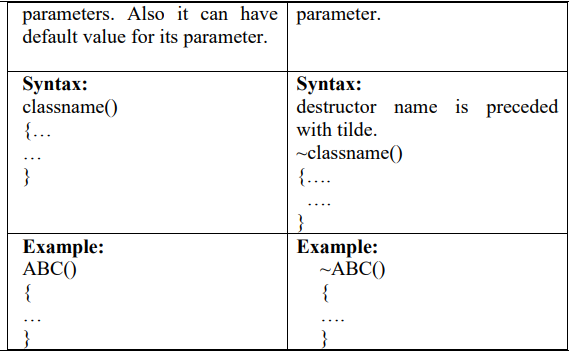
The memory space for object is allocated when they are declared and not when the class is specified. The member functions are created and placed in memory space only once when they are defined as a part of a class definition.

Since all the objects belonging to that class use the same member functions, no separate space is allocated for member functions. When the objects are created only space for member variable is allocated separately for each object. Separate memory locations for the objects are essential because the member variables will hold different data values for different objects



**9. Difference between: Constructor and Destructor (S-19) (4M)**

****

****

**10. Write a C++ program to create class STUDENT. The data members of student class: Roll\_no,name, department. Declare a parameterized constructor with default value for department as 'CO'to initialize members of object. Initialize and display data for 2 students. (S-19) (6M)**

#include<iostream.h>

#include<conio.h>

#include<string.h>

class student

{

int roll\_no;

char name[20],department[40];

public:

student(int rno,char \*n,char \*d="CO")

{

roll\_no=rno;

strcpy(name,n);

strcpy(department,d);

}

void display()

{

cout<<"\n Roll No:"<<roll\_no;

cout<<"\n Name:"<<name;

cout<<"\n Department:"<<department;

}

};

void main()

{

student s1(112," Chitrakshi"),s2(114,"Anjali");

clrscr();

s1.display();

s2.display();

getch();

}

**11. What is class? Give its example (W-19) (2M)**

Class is a user defined data type that combines data and functions

together. It is a collection of objects of similar type.

Example:

class Student{

int rollno;

char name [ I O];

public:

void getdata( );

void putdata( );

};

**12. Write 2 properties of static member function (W-19) (2M)**

i) A static member function can have access to only other static data members and functions declared in the same class.

ii) A static member function can be called using the class name with a scope resolution operator instead of object name as follows: class\_name::function\_name;

**13. State rules for writing destructor function (W-19) (4M)**

1) A destructor is a special member function which should destroy the objects that have been created by constructor.

2) Name of destructor and name of the class should be same.

3) Destructor name should be preceded with tilde (~) symbol.

4) Destructor should not accept any parameters.

5) Destructor should not return any value.

6) Destructor should not be classified in any types.

7) A class can have at most one destructor.

**14. What is parameterized constructor? (W-19) (4M)**

A constructor that accepts parameters is called as parameterized constructor.

In some applications, it may be necessary to initialize the various data members of different objects with different values when they are created. Parameterized constructor is used to achieve this by passing arguments to the constructor function when the objects are created.

Example:

class ABC{

int m;

public:

ABC(int x)

{

m=x;

}

void put()

{

cout<<m;

}};

void main(){

ABC obj(10);

obj.put();}

In the above example, constructor ABC (int x) is a parameterized constructor function that accepts one parameter. When “obj” object is created for class ABC, parameterized constructor will invoke and data member m will be initialized with the value 10 which is passed as an argument. Member function put ( ) displays the value of data member “m”.

**15. Explain friend function with proper example (W-19) (4M)**

The private members of a class cannot be accessed from outside the class but in some situations two classes may need access of each other’s private data. So a common function can be declared which can be made friend of more than one class to access the private data of more than one class. The common function is made friendly with all those classes whose private data need to be shared in that function. This common function is called as friend function.

Friend function is not in the scope of the class in which it is declared. It is called without any object. The class members are accessed with the object name and dot membership operator inside the friend function. It accepts objects as arguments.

[6:59 PM, 10/14/2022] Azlan: Example:

Program to interchange values of two integer numbers using

friend function.

#include<iostream.h>

#include<conio.h>

class B;

class A

{

int x;

public:

void accept()

{

cout<<"\n Enter the value for x:";

cin>>x;

}

friend void swap(A,B);

};

class B

{

int y;

public:

void accept()

{

cout<<"\n Enter the value for y:";

cin>>y;

}

friend void swap(A,B);

};

void swap(A a,B b)

{

cout<<"\n Before swapping:";

cout<<"\n Value for x="<<a.x;

cout<<"\n Value for y="<<b.y;

int temp;

temp=a.x;

a.x=b.y;

b.y=temp;

cout<<"\n After swapping:";

cout<<"\n Value for x="<<a.x;

cout<<"\n Value for y="<<b.y;

}

void main()

{

A a;

B b;

clrscr();

a.accept();

b.accept();

swap(a,b);

getch();

}

**16. Write a program to declare a class 'student' having data members as stud\_name, roll\_no Acceptand display data for 5 nos. (W-19) (6M)**

#include<iostream.h>

#include<conio.h>

class student

{

int roll\_no;

char stud\_name[20];

public:

void Accept();

void Display();

};

void student::Accept()

{

cout<<"\n Enter student‟s name and roll no\n";

cin>>stud\_name>>roll\_no;

}

void student::Display()

{

cout<<stud\_name<<”\t”<<roll\_no<<”\n”;

}

void main()

{

student S[5];

inti;

clrscr();

for(i=0;i<5;i++)

{

S[i].Accept();

}

cout<<”Student details \n Student‟s Name \t Roll No\n”;

for(i=0;i<5;i++)

{

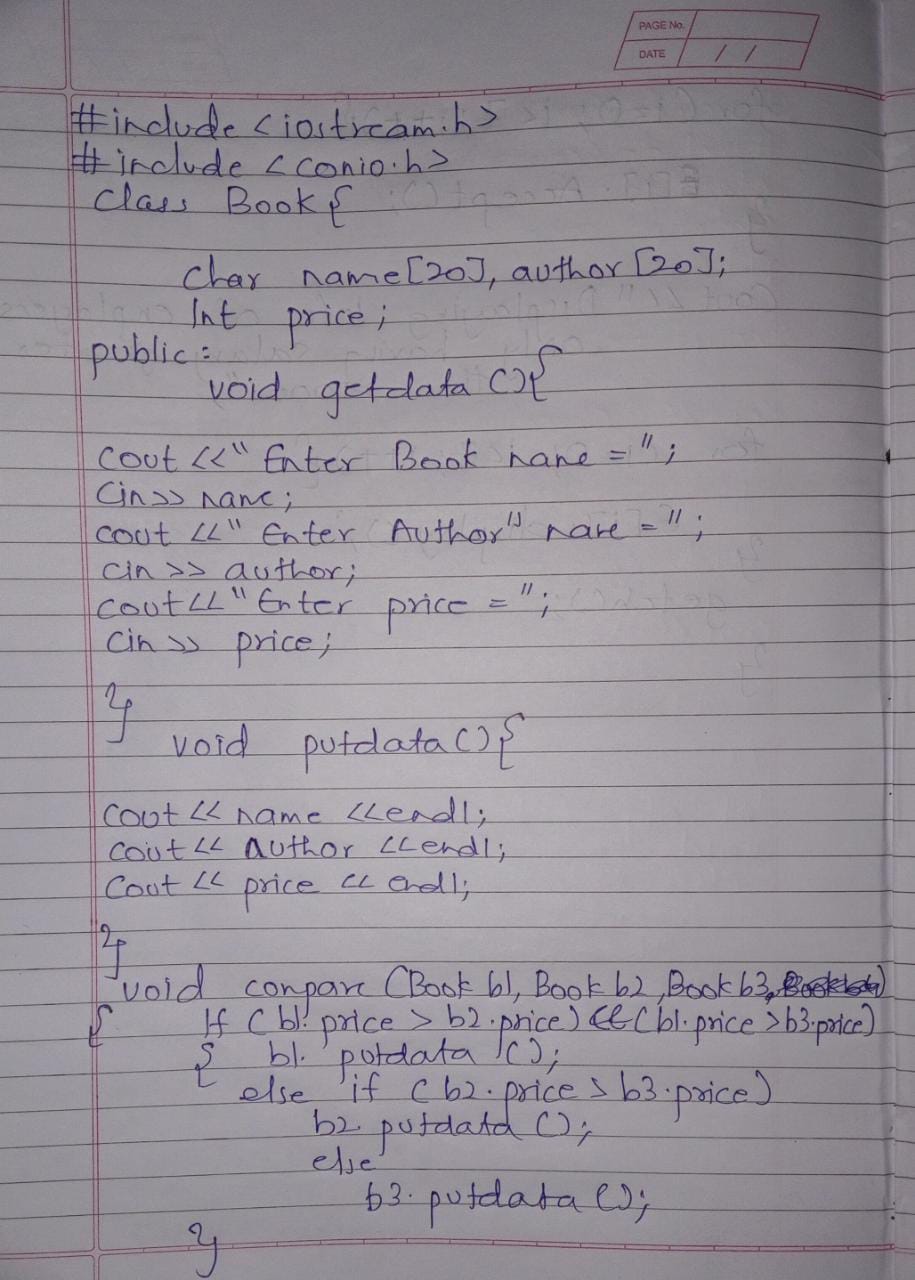
S[i].Display();

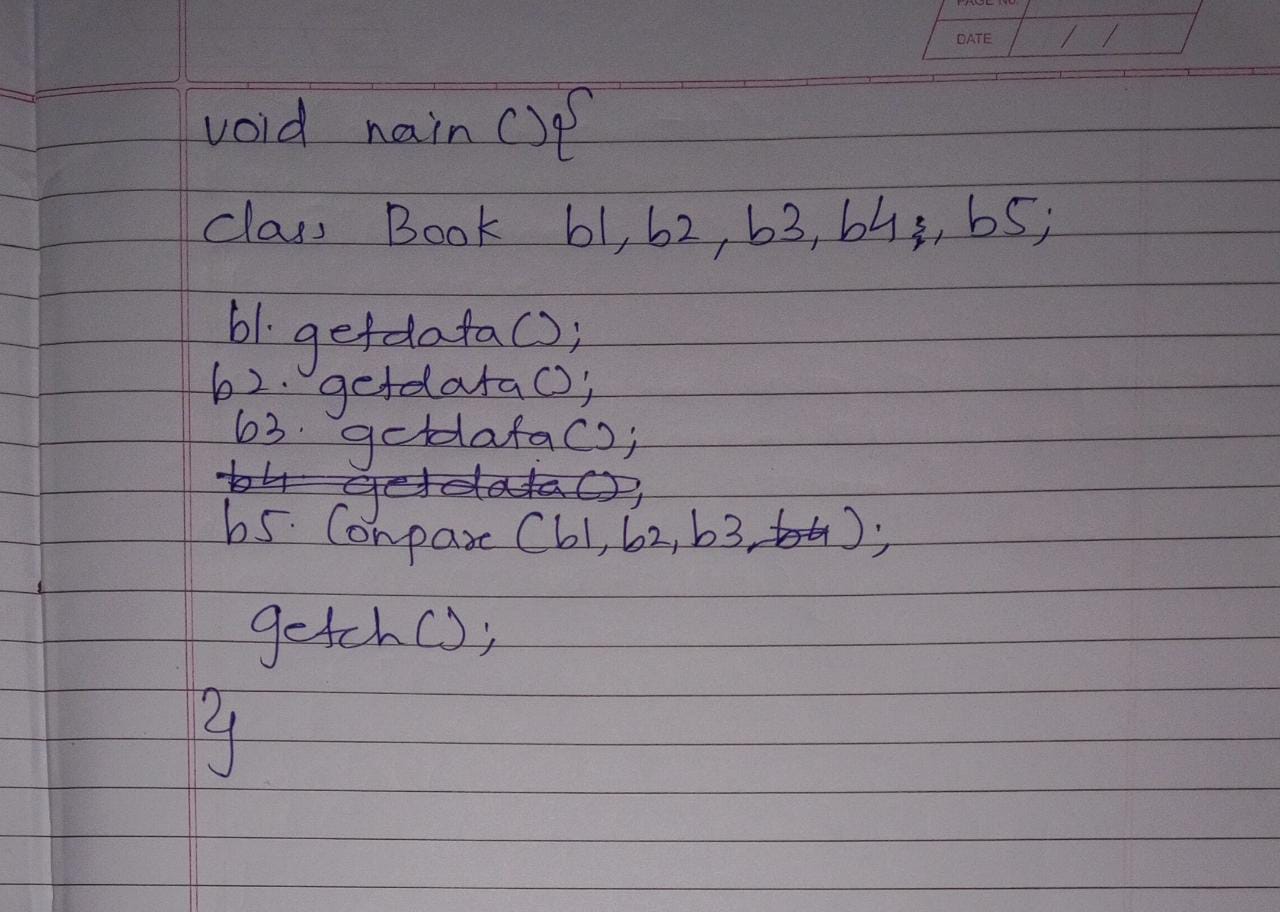
}

getch();

}

**17. Write a C++ program to declare a class 'Book' having data members book name, author and price. Accept and display data for book having maximum price.**

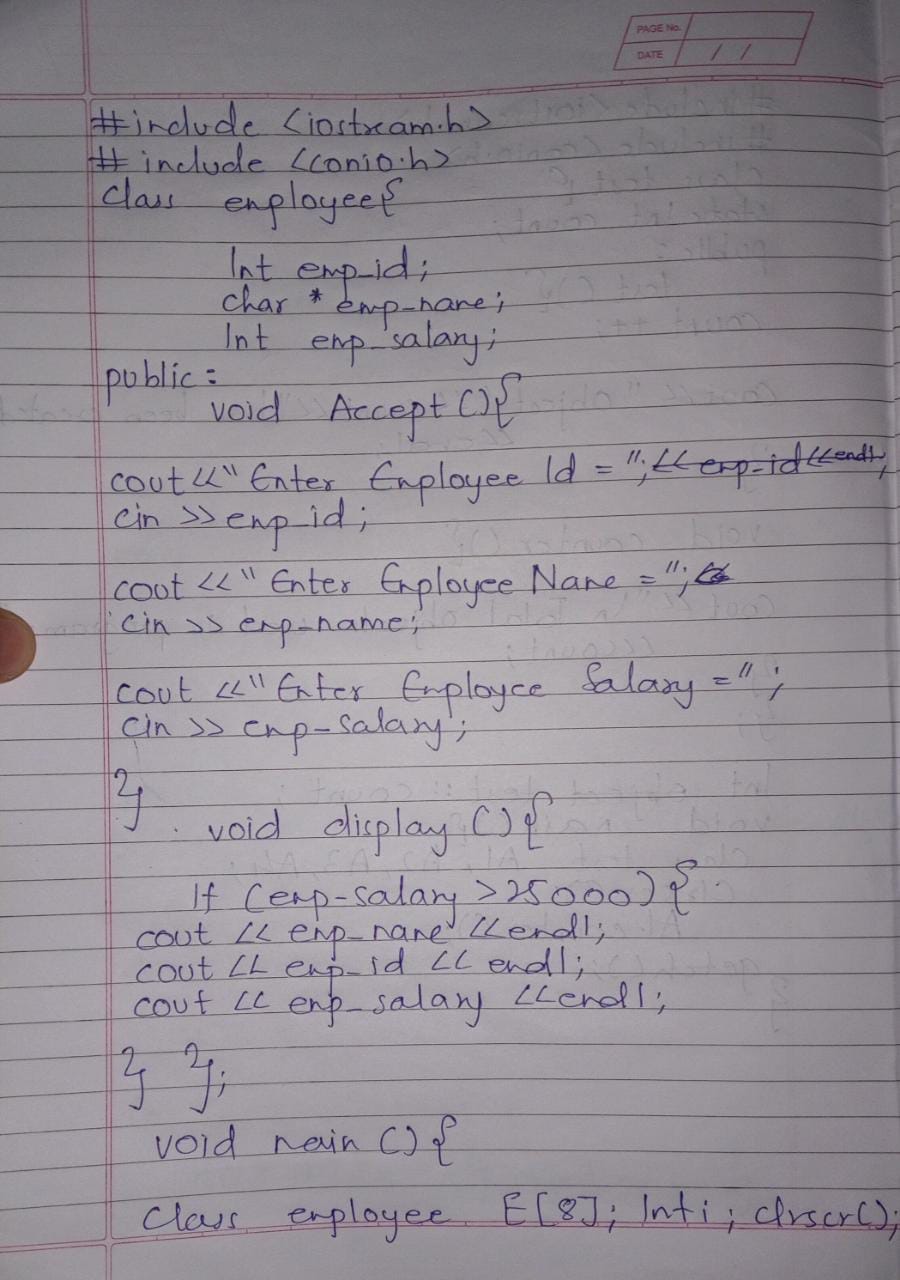
****

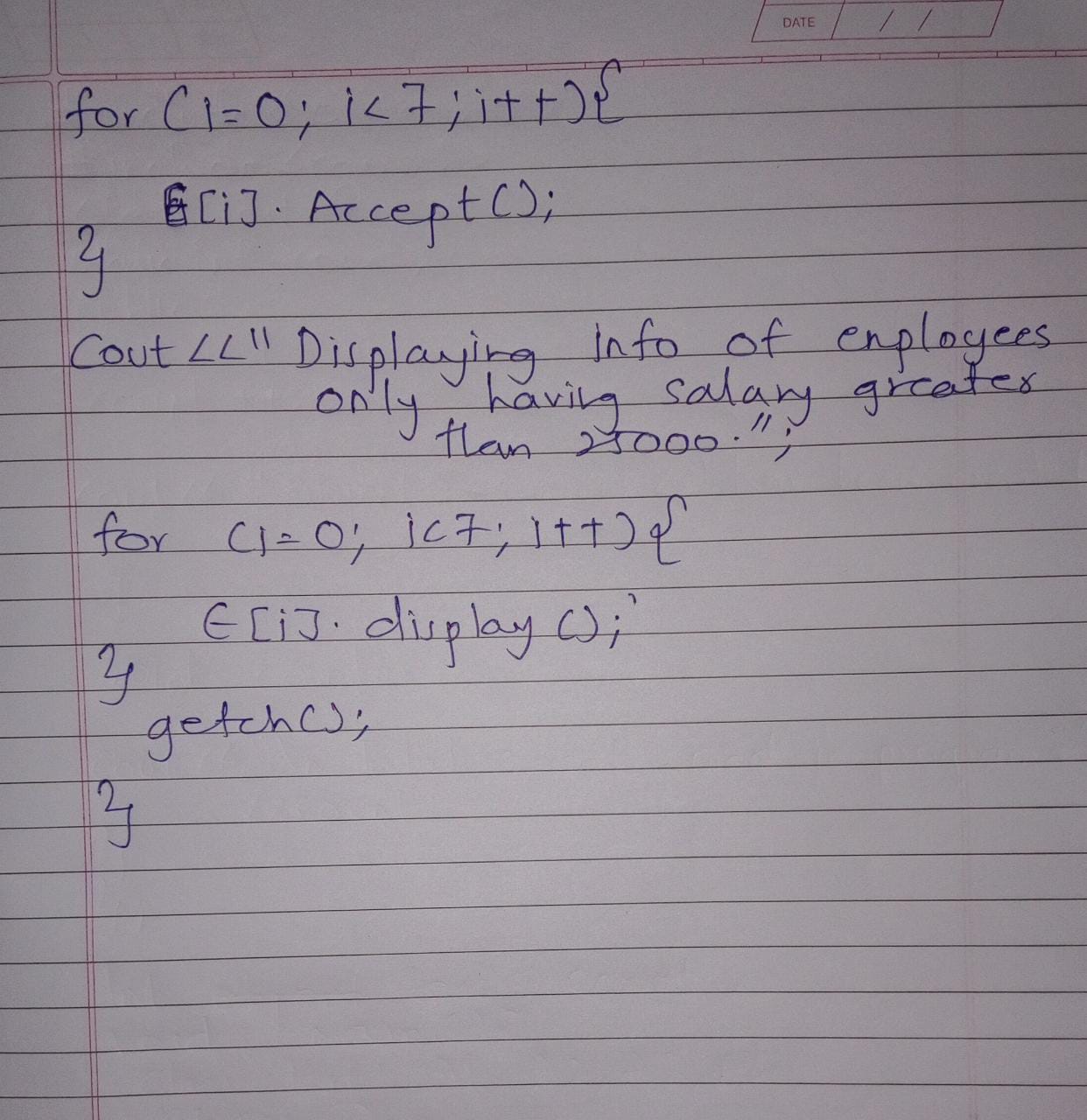
****

**18. Write a C++ program to declare a class 'staff having data members name, basic salary, DA, HRA and calculate gross salary. Accept and display data for one staff. Where DA=74.5% of basic,HRA=30% of basic and gross**

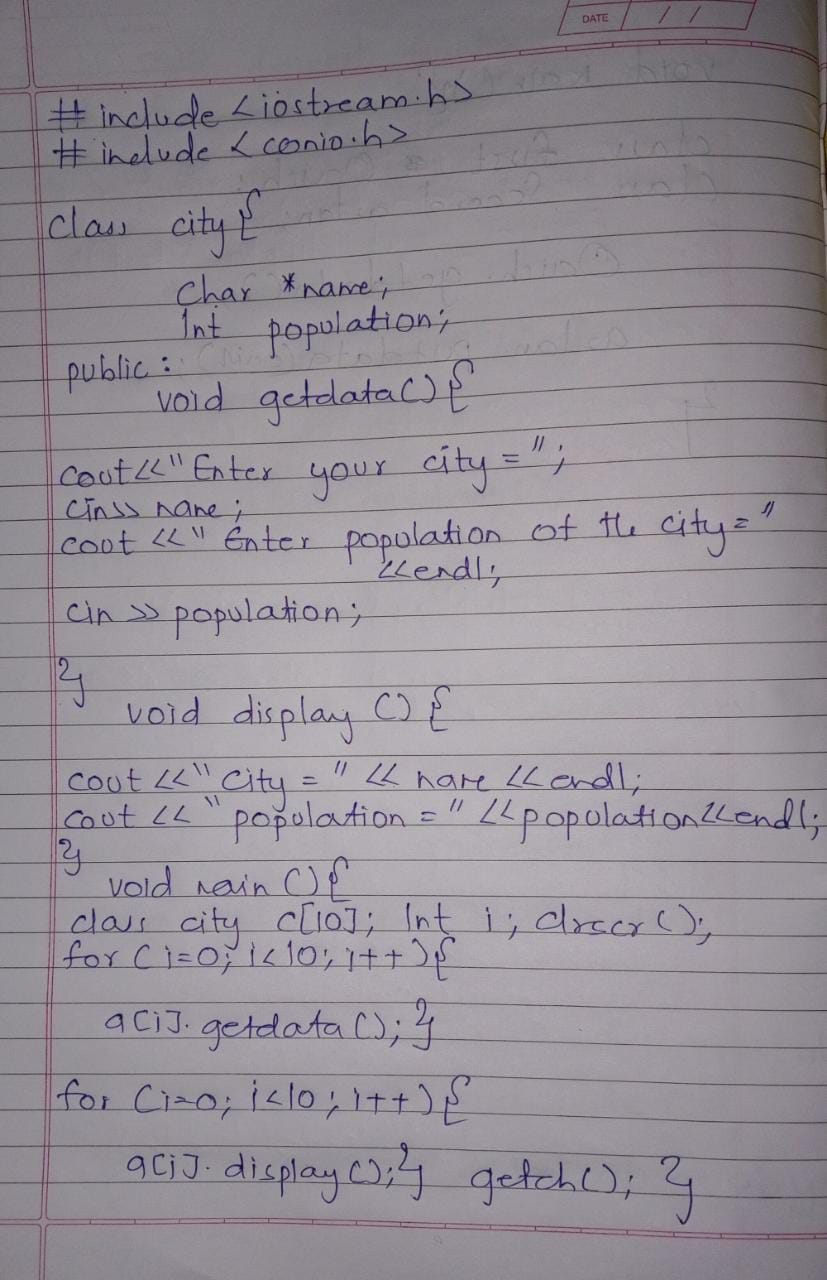
**salary=basic+HRA+DA**

**19. Write C++ program to define class employee having data members emp\_id, emp\_name and emp\_salary. Accept and display data for employees having salary greater than 25000/-.**

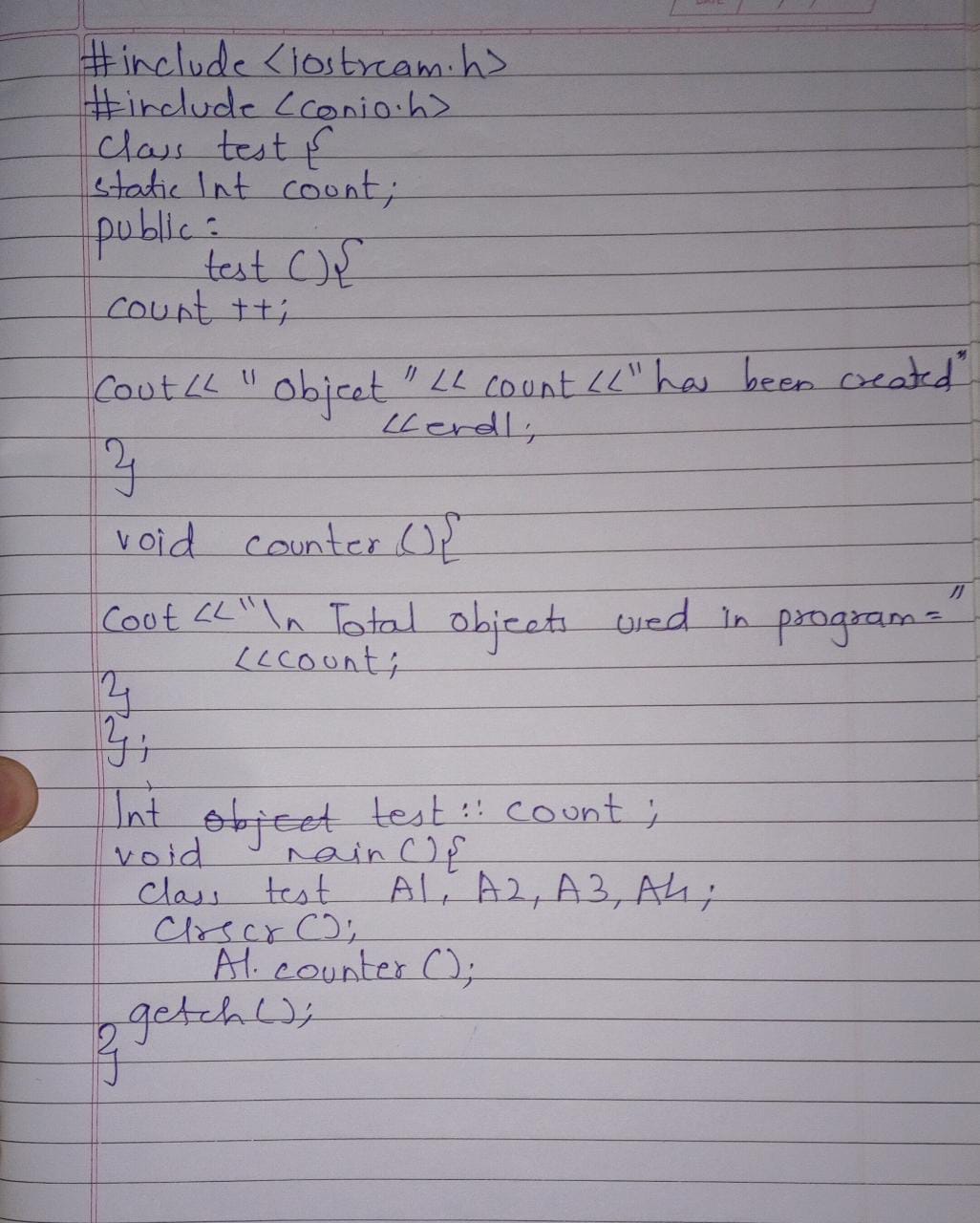
****

****

**20. Write a C++ program to define a class 'City' having data members city name, population. Accept and display data for 10 cities.**

****

**21. write a program to declare class test having data member as count. Program which will print number of objects created by user.**



**22. Explain inline function with an example**

**CHP 3:**

**1. List different types of inheritance with diagram**

