10.1

Page No.
Date :

This program uses multiple printf statements to write or print christmas tree. Printf is a function that prints formatted text to the standard output stream and is included in <stdio.h> header file.

Syntax:

printf (string format, items-to-format)

is a string that includes text to be printed literally and marks to be % replaced by the text obtained from the additional parameters

These may be variables or literals.

Format specifiers — contains a % character followed by a letter to indicate what type of value will be formatted there. For example

%c — character or char
%s — string of characters
%d — Integer
%f — float, etc..

'\n' is used to enter a line break.

Scanned with OKEN Scanner

1.2

Page No.
Date :

Approach 1: User enters a number, which is matched for its corresponding month name using a switch block with multiple cases.

Approach 2: This approach uses a ~~less~~ lesser number of lines of code than the approch I. This program stores all month names in an array and accesses them via their indices.

For example:

User enters 5, then the ~~non~~ month name is monthNames[4]

Scanned with OKEN Scanner

1.3

Page No
Date :

There are various input statements used in C with different use cases. getchar() is used to get a single character from standard input and returns the char. scanf() is a more general function which is used to get any type of input form standard input, may it be an integer, float, char or string. In this example a character is being read from standard input. getch() is another function which is typically used to hold the program onto console until any key is pressed.

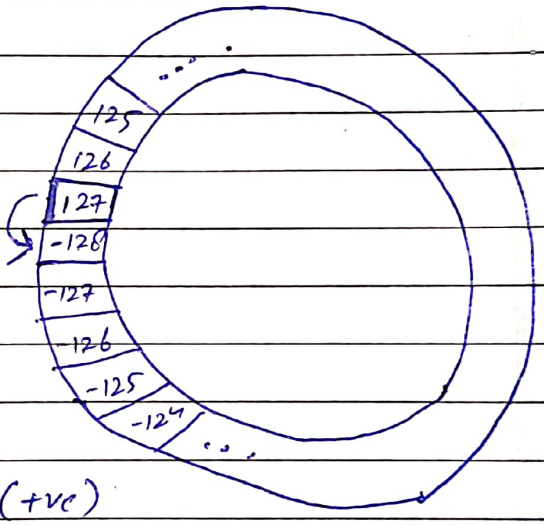
Scanned with OKEN Scanner

1.4

Page No.
Date :

Every datatype in C has a specified size and hence it can store values in a given range only. For example size of char is 1 byte so its range is from -128 to 127. What will happen if we ~~want~~ to try to store +128 in a char variable.

```
char ch = 128;
printf("%c", ch);   // prints -127
```



The values form a kind of logical ring where the 128 ends meet. If 128 is tried to be stored onto a char then it is one more than the range (+ve) so it jumps to -128.

The same behavior is shown by all other primitive data types.

Scanned with OKEN Scanner

2.1

Page No.
Date :

Operator precedence determines which operation is performed first in an expression with more then ~~one~~ one operator with different precedence. For example.

$7 + 3 * 2$

$7 + 6$

13

multiplication has higher precedence over addition.

$15 / 5 * 2$

3

6

Both * and / have same precedence value but associativity is from left to right hence 15/5 is evaluated prior to 5 * 2.

2.2

Page No.
Date :

The program uses two different loops for printing numbers in an ascending order and descending order.

The first loop starts from 0 and ends at 19, printing the number on each iteration

0, 1, 2, 3, 4, 5, ..., 16, 17, 18, 19

The subsequent loop does the complete opposite starting with 20 and ends on 0 printing the number on each iteration:

20, 19, 18, 17, 16, ..., 5, 4, 3, 2, 1, 0

Syntax of for loop

executes for the first time

Executes n+1 times

Executes n times.

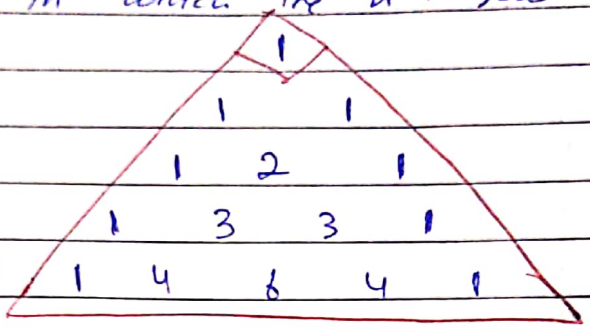
```
for (initialization; condition; operation)
{ ... }
```

2.3

Page No.
Date :

Pascal's triangle is a triangular array of binomial co-efficients in which the n^{th} row contains binomial coefficients

$^nC_0, {}^nC_1, {}^nC_2, \ldots {}^nC_n$



^nC_r can be represented as $C(n,r)$ and this represents the n^{th} row's r^{th} element. The idea is to calculate $C(n,r)$ using $C(n,r-1)$. It can be calculated in $O(1)$ time using the following formula:

$$C(n,r) = \frac{C(n,r-1) * (n-r+1)}{r}$$

Page No.
Date :

Reversing a number

For example: n = 5406

rev_num = 0

Last digit —— 6

rev_num
rev_num * 10 + 6
= 6

Last digit = 0 — 6 * 10 + 0 = 60

= 4 — 60 * 10 + 4 = 604

= 5 — 604 * 10 + 5 = 6045

The idea is to get the last digit of the number and increase the place value of reverse number by 10 and add the digit to it.

Scanned with OKEN Scanner

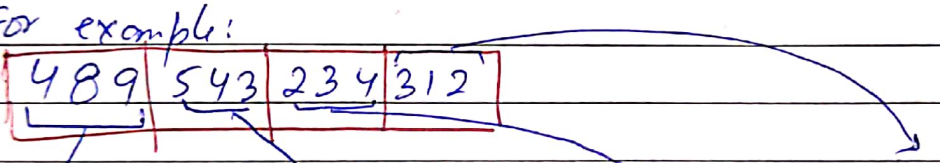
3.1

Page No.
Date :

I have created a function threeFigureWord() which takes as an input the number which can't have more than 3 digits and returns the string (words) of that number. For example 123 — "one hundred twenty three".

Then there is another function toWords() which first checks a number if it is in billions gets the 3-digit word and appends billions to it. the same routine applies to the number until it is completely resolved.

For example:



four hundred eighty nine + billion
five hundred forty three + millions
two hundred thirty four + thousands
three hundred twelve

"Four hundred eighty nine billions five hundred fourty three millions two hundred thirty four thousands three hundred twelve"

Scanned with OKEN Scanner

3.2

Page No.
Date :

isPrime (int) is a function which takes as an argument a number and returns TRUE if the number is prime else FALSE.

The program loops over all the number in the range [100, 500] and sums all those numbers, which are prime, in a variable called sum.

Scanned with OKEN Scanner

Page No.
Date :

char Str[100] ⟶ Allocated a contiguous memory of 100 bytes and can be used to store characters.

gets(str) :

Reads an input string from the standard input and writes it back to that contiguous array that was allocated by char Str[100];

Scanned with OKEN Scanner

3.4

Page No.
Date :

User enters 20 randomnly chosen number and these numbers are stored in an array called nums.

A 2D array is used to calculate the frequency of each element. The first logical column stores the element and another stores the frequency of that respective number.

A for loop iterates over all array elements and checks for its presence in the frequency array:

- If the element is found, the frequency is incremented
- otherwise, The element is added into the frequency table with values as 1.

Scanned with OKEN Scanner

4.1

Page No.
Date :

Removing duplicates from an array in-place requires the index or size of array to change. If there are 5 elements in an array and 2 elements are same then the array after removing duplicates from it will have 4 as its size.

Example:

index	10	10	10	30	40	40	50	80	80	100
0	10									
1										
2										
3		30								
4			40	50						
5										
6										
7					80					
8										
9						100				

10	30	40	50	80	100

New size of array is 6.

Scanned with OKEN Scanner

4.2

Page No.
Date :

Store all positive elements in a temporary array and shift all negative elements to right while preserving there order.

After the negative elements are in their final order, start overwriting the positive numbers from temp array from left. Finally the program arrives to the desired state.

Example:

10	-15	1	3	-2	0	-2	-3	2	-9

Temp array = [10, 1, 3, 0, 2]

Shifting negative elements to right :

10, -15, 1, 3, -2 | -15, -2, -2, -3, -9

overwriting positive elements

10,	1,	3,	0,	2,	-15,	-2,	-2,	-3,	-9

Desired state.

Scanned with OKEN Scanner

4.3

Page No.
Date :

Let matrix1 and matrix2 be two 3×3 matrices. The addition matrix is calculated by summing up matrix1[i][j] and matrix2[i][j] elements for M[i][j]

$$\therefore \text{addition}[3][3] = \begin{bmatrix} 1+2 & 2+2 & 3+5 \\ 5+5 & 6+4 & 7+1 \\ 9+1 & 4+1 & 2+7 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 8 \\ 10 & 10 & 8 \\ 10 & 5 & 9 \end{bmatrix}$$

Change every i,j th element to j,i th element and you have a transpose. For example the transpose of matrix1 is :

$$\text{transpose}[3][3] = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 4 \\ 3 & 7 & 2 \end{bmatrix}$$

Multiplication : [i][j] is the summation of multiplication of the respective row element with the column element.

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 6 & 7 \\ 9 & 4 & 2 \end{bmatrix}_{3\times3} * \begin{bmatrix} 2 & 2 & 5 \\ 5 & 4 & 1 \\ 1 & 1 & 7 \end{bmatrix}_{3\times3} = \begin{bmatrix} 2+10+3 & 2+8+3 & 5+2+21 \\ 10+30+7 & 10+24+7 & 25+6+49 \\ 18+20+2 & 18+16+2 & 45+4+14 \end{bmatrix}$$

$$\Downarrow$$

$$\begin{bmatrix} 15 & 13 & 28 \\ 47 & 41 & 80 \\ 40 & 36 & 63 \end{bmatrix}$$

Scanned with OKEN Scanner

4.4

Page No.
Date :

funs.h - header files contains the declaration of the function printArr. funs.c contains the definition, and is used in another file called main.c.

The two files funs.c and main.c are compiled seperately and are linked together using commands.

gcc -c funs.c compiles funs.c file and creates an object file called funs.o. Similar approach is applied to main.c

gcc -c main.c

gcc main.o funs.o -o a

Two object files are linked together and a program is generated which is designated by a.exe

./a executes the program.

5.1

Page No.
Date :

struct Student* setStudent(struct Student* std, int id, char* name, float percentage);

is a function which takes as argument the pointer of a pre-defined Student object and changes the member values by the arguments provided.

```
setStudent(&std, 10, "Ovais Ahmed", 93.01f);
```

will change id of the student std to 10, name to "Ovais Ahmed" and marks to 93.01f.

Scanned with OKEN Scanner

Page No.
Date :

The program starts with opening of a file if the file is not present it dies create a file with name "w05_p2.txt".

```
$ gcc w05_p2_cli_in_files.c
```

will compile this file

```
$ ./a 4 "Ovais Ahmad Khanday" "University of Kashmir" MCA B-2024
```

All command line arguments are stored in argv[] array and argc is the total number of arguments.

`fprintf(fp, "%s\n", argv[i])` adds each argument on a new line in w05-p2.txt file.

5.3

Page No.
Date :

malloc() is a function that allocates a contiguous block of memory and it is included in stdlib.h header file.

The program starts by asking a user to enter the number of students he/she wants to store. Using malloc repeatatively the storage for each student is allocated and the pointer is stored in an array.

After the array of students is completely allocated user heads onto entering details of each student.

The details are printed of each student. Now it is the right time to free the allocated space. This is done using free() function on each element of the pointer array.

free(stds[i])	frees/deallocates the space of the student object at location stds[i].

Scanned with OKEN Scanner

5.4

Page No.
Date :

Stubb ( ) function adds a string to another string at some location and deleting some characters from destination string.

An auxiliary character string is created which can accomodate both strings.

The process starts with adding first [sp] character to the auxiliary space and then adding the whole second string, at the position.

ALGORITHM

1. Create result array of size $s_1 + s_2$
2. Add first sp characters of s_1 into result
3. Keeping the same index at beginning, add s_2 to result.
4. Adding the remaining [sp + rp, s_1) characters to result.

6.7

Page No.
Date :

User enters details of each person and Then the array is passed to a function which sorts the persons according to the name as a primary key and address as a secondary key.

The sort function firstly sorts the array as per names of persons only. Then sorts the persons according to addresses. If the name of more than 1 person is same then they are swapped as per address. For example:

```
Enter number of persons: 4

Enter name for (1): Ovais
Enter address for (1): Anantnag
Enter telephone for (1): 2323232323

Enter name for (2): Abass
Enter address for (2): Kulgam
Enter telephone for (2): 2223334456

Enter name for (3): Ovais
Enter address for (3): Aang
Enter telephone for (3): 3453453454

Enter name for (4): Abass
Enter address for (4): Anantnag
Enter telephone for (4): 22334454322

Abass, Anantnag, 2233445432
Abass, Kulgam, 2223334456
Ovais, Aang, 3453453454
Ovais, Anantnag, 2323232323
```

Scanned with OKEN Scanner

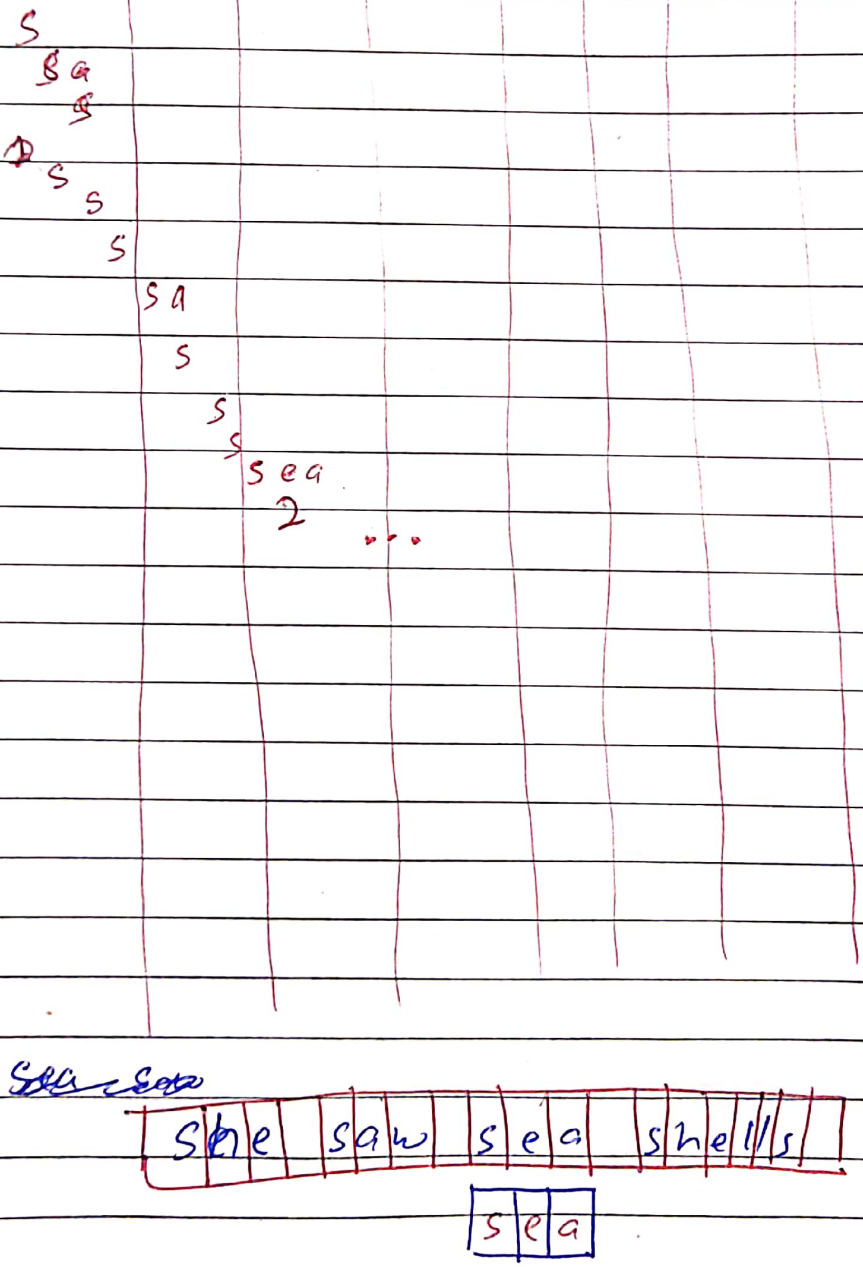
6.2

Page No.
Date :

getCount( ) functions implements the Naive approach of finding the occurrences of a word in a string. If the size of string is n and pattern is m the time complexity is O(m.n) and space complexity is O(1).

sea

She saw sea shells on the sea shore



Scanned with OKEN Scanner

6.3

Page No.
Date :

Assumming that the destination string can accomodate the total size after adding another string to it.

```
while (*currentDest) currentDest*
```

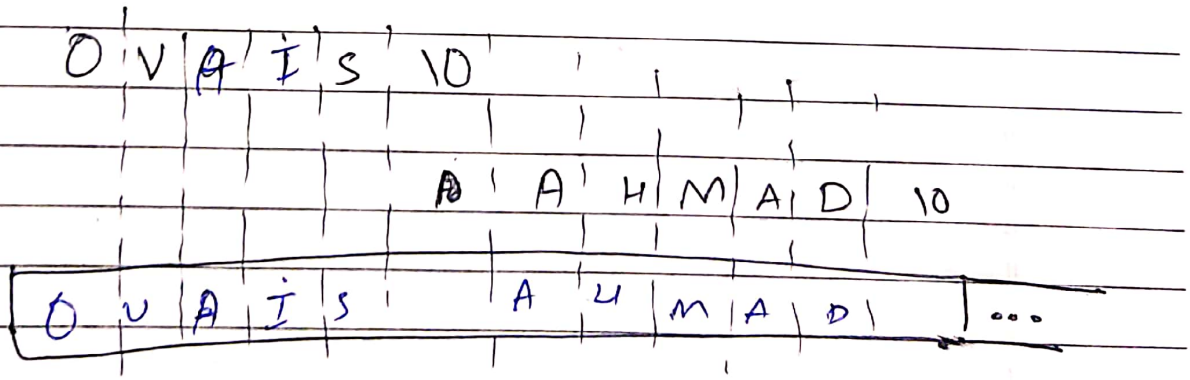
takes the pointer to null character

```
while (*currentDest = *src) {
    src++;
    currentDest++
}
```

keep on adding all character from src string to destination until '\0' (NULL) is encountered in src string.

Example:

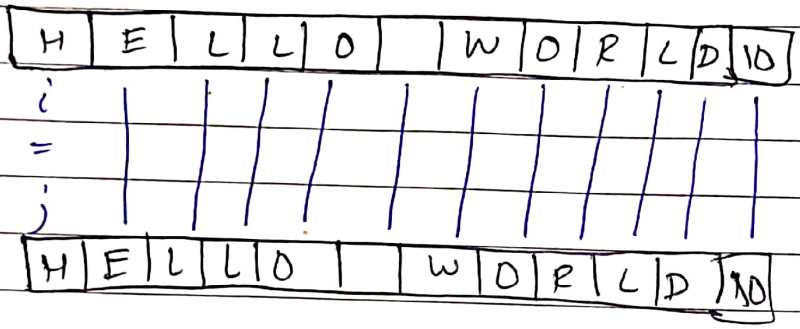
```
char str[100] = "OVAIS";
char str1[] = "AHMAD";

concat(str1, str2);
```

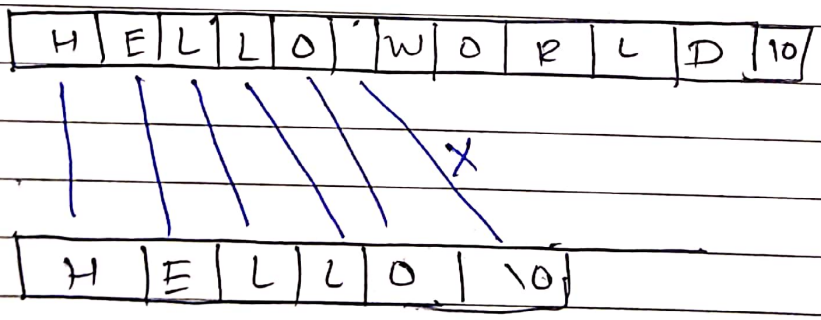


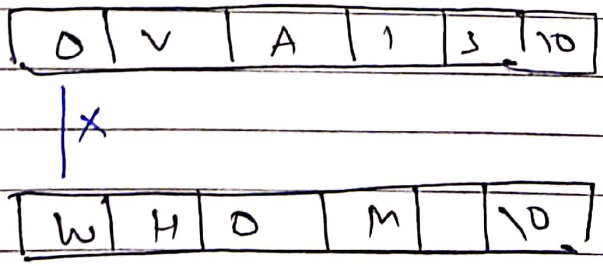
6.4

Page No.
Date :

Using two pointer algorithm — one pointer on initial character [0] of string 1 and another pointer on initial character [0] of second string. If both pointers traverse the string while keeping the condition valid (equal) and encounter both '\0' (NULL) then the strings are equal.



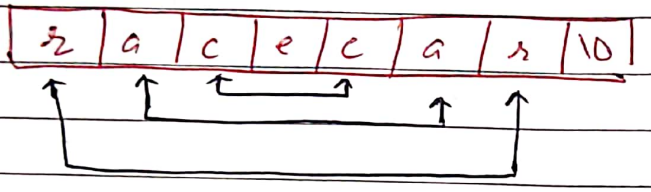




6.5

Page No.
Date :

Again this program uses two pointers algorithm. one pointer from front and another from rear. If both values are same then continue till middle else if anywhere 2 values are not equal then the string is not a palindrome.



6.6

Page No.
Date :

Traverse the array — check if the current character is vowel, (heuristic — vowels are less in number — more optimized solution is possible), then increment the vowels count else if the current character is not whitespace increment consonant count.

Assumption: All but vowels and space is a consonant.

Example.

You were spring
And I the edge of a cliff
And a shinning waterfall rushed over me

Vowels = 25
Consonants = 42.

isVowel() function takes as an argument a character and returns true if it is any one of the following.

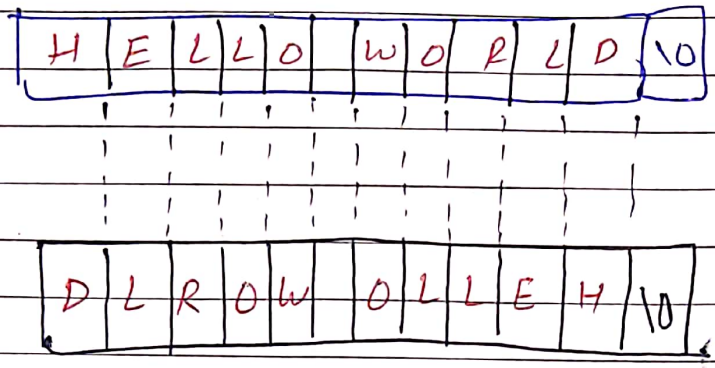
'a', e, i, o, u, A, E, I, O, U

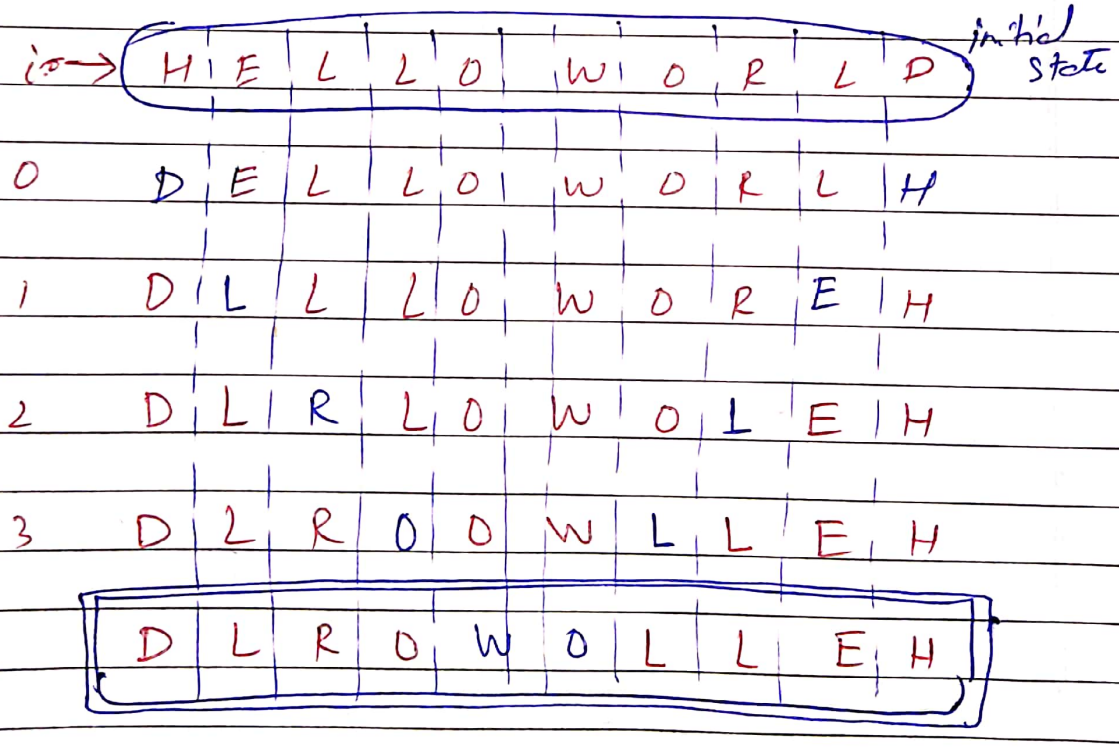
7.1

Page No.
Date :

Reversing a string in-place requires swapping of i^{th} and $n-i^{th}$ character. This problem can be solved using two pointer algorithm.

For example:



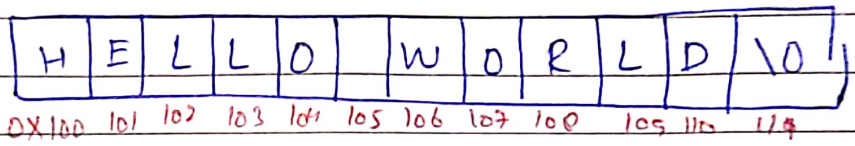


7.2

Page No.
Date :

Array is a contiguous block of elements of a some type. If we have the address of an element we can traverse back and forth by decrementing and incrementing the address. This is demonstrated below:

char *str = "Hello world";



*(str+0) = *0X100 ——— 'H'

*(str+1) = *0X101 ——— 'E'

*(str+2) = *0X102 ——— 'L'

*(str+3) = *0X103 ——— 'L'

*(str+4) = *0X104 ——— '0'

Scanned with OKEN Scanner

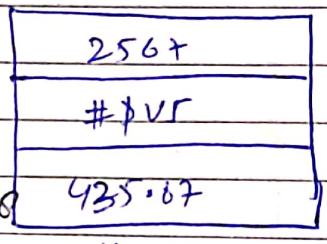
7.3-

Page No.
Date :

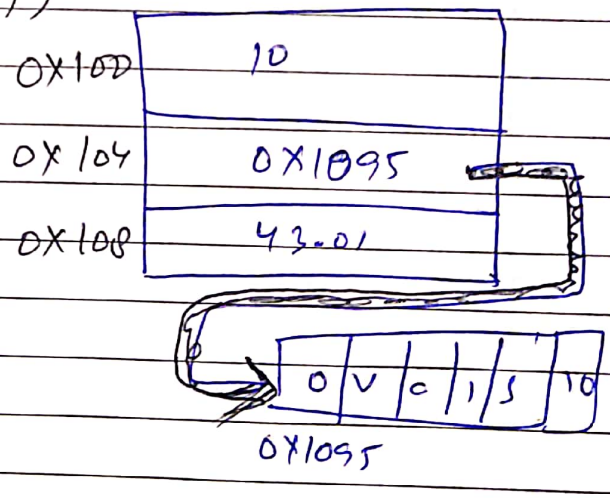
```
Student* fill (Student* std, int id, char* name, float marks){
    std->id = id;
    std->name = name;
    std->mark = marks
    return std;
}
```

This function takes pointer to the student object and add, id, name, marks to the student object accessing it via address.

```
int main(){
    Student std; //creat a block of memory with GV;
```




```
fill(&std, 10, "Ovais", 43.01f)
```



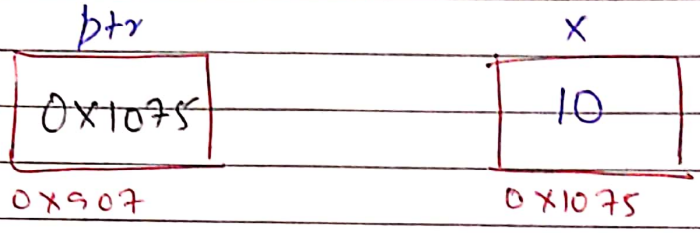
Scanned with OKEN Scanner

8.1

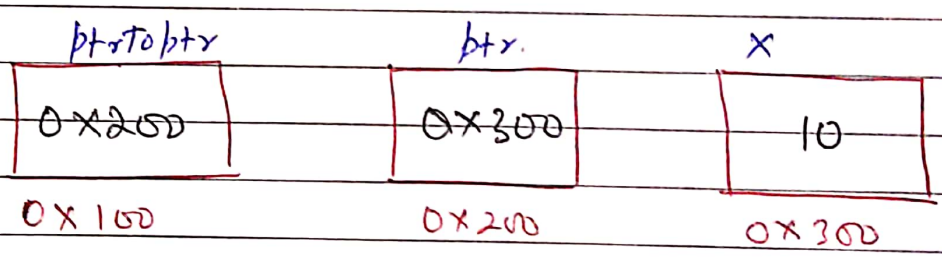
Page No.
Date :

A pointer is a variable that stores the address of an object in memory. The object can be accessed via the memory location, which is stored in the pointer.

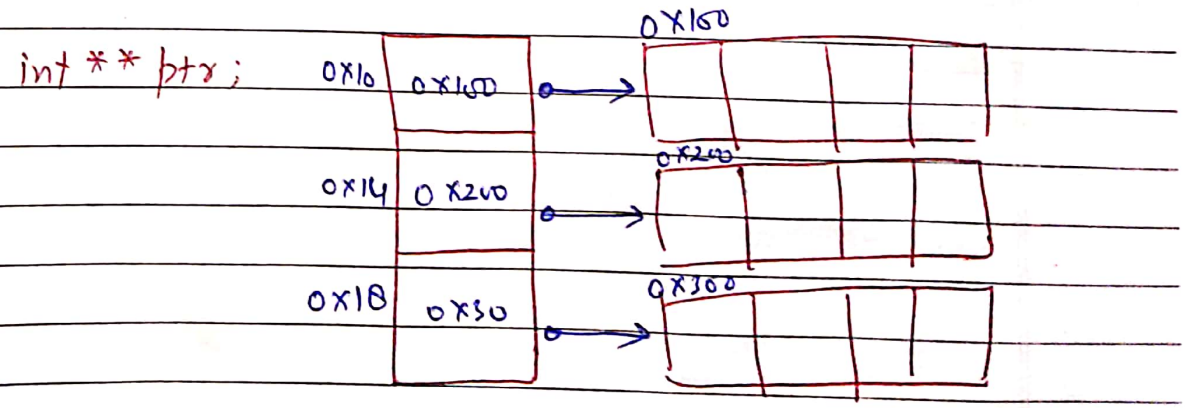
A pointer-to-pointer is a type of pointer which stores the address of another pointer that stored the address of the actual object.



int *ptr = &x;



int **ptrtoptr = &ptr;



Scanned with OKEN Scanner

8.2.

Page No.
Date :

Unlike normal pointers, a function pointer points to code, not data. Typically a function pointer stores the start of executable code.

```
int (*funPtr)(int, int) = sum
```

funPtr is a pointer to sum function. The sum function can be called in two ways

1. sum(2,4), and
2. funPtr(2,4), because both point to the same block of executable code.

8.3.

Page No.
Date :

Swapping of object fields using pointers:

Two objects are passed into a function by their addresses, Now whatever changed are made, will be reflected in the actual object.

The program uses another temporary object that stores values of first object. The first object is then overwritten by the second, and restored by temporary.

8.4

Page No.
Date :

Classes are used to define data-types by users. These data-types can be combination of any number of primary or user-defined data-types.

Complex class has 2 member attributes. As every complex number has real and imaginary part, we have defined 2 floating numbers one for each.

Complex(float r, float i) is a constructor which is used to initialize a new object.

```
Complex c1 = Complex(29, 20);

Complex c2 = {10, 12.5f};

Complex* c2 = new Complex(11, 22.011f);
```

are the 3-ways we can instantiate new objects of any class (complex here).

8.5

Page No.
Date :

Person is a user defined data-type which has some attributes such as: name, age, address, and height.

```
class Person {
    String name;
    int age;
    String address;
    float height;

    public:
    Person(String n, int a, String ad, float h) {
        name = n;
        age = a;
        address = ad;
        height = h
    }
    void print();
}
```

Attributes — private by default

Constructor

used to initialize new objects while taking as input all the fields required.

9.1

Page No.
Date :

There are two classes defined in this code, Person and Complex each having its own members pertinent to them. In-order to create objects of the classes defined we can write the code as follows:

```
Complex c = Complex(10.0f, 12.0f);
```

Class Name

Constructor

object name

Assignment operator

Syntax:

```
class_name object_name = class_name(...args);
```

Another Syntax

```
class_name obj_name = {...args};
```

Page No.
Date :

Both classes have a print() function defined that can access the private members of that particular class and print them in the format specified/defined inside the function.

The functions need to be public, in order to be available to main() or any where else. We can simply call the function after creation of an object of any of the two classes is completed. For example:

```
Complex c = Complex(10.0f, 12.0f);
```

c.print(); ⟶ print() is a public function and prints the member attributes to Stdin.

Page No.
Date :

Scope Resolution Operator :: — is an operator

which is used for multiple purposes. One of the most useful and used purpose is, that it defines a function outside a class helps

For example

```
void print();  // Function is only declared here inside the class complex.
```

```
void Complex::print() { - - - }
```

↓ Class_name:: is prefended to distinguish that the function belongs to a particular class and if there are any scoping conflicting, those gets resolved.

Scanned with OKEN Scanner

9.4

Page No.
Date :

Constructor - are methods used to instantiate new objects of a class. There are 3-types of constructors:

- Default Constructor — Defined with no arguments
- Overloaded constructor — may have any #number of args (>0)
- Copy constructor — Takes as I/p an object and returns new object with same attribute values.

```
Student() {                          ——— No argument
    age = 18
    marks = 33.0f;                   } sets some default values in most of the cases
}
Student(String name, int age, float marks) {
    this->name = name;               ]
    this->age = age;                 ] All or some arguments are passed
    this->marks = marks;             ]
}
Student(Student &p) {
    this->name = p.name;             ]
    this->age = p.age;               ] A reference to an object is passed as an argument and
    this->marks = p.marks;           ] a new object with same member attributes is created and returned.
}
```

10.1

Page No.
Date :

Access specifiers - Are keywords used in class definition which define the access to members. There are 3 access specifiers in C++

- public ——— accessed from outside and inside classes
- protected ——— accessed from all inherited classes
- private ——— accessed from the same class members.

Person ——— name, age, phoneNo

protected.

Can be accessed from student class but not from main().

Student ——— rollNo, marks

private

These members can't be accessed anywhere but Student class.

Person ——— Person(...).
Student ——— Student(...), print()

Public

These constructors and member functions are publicly accessed from anywhere in the code.

10.2

Page No.
Date :

An inline function defines a block of code with a name, whenever and wherever the name is used - it is replaced / populated by the actual lines of code at preprocessing or compile-time.

```
inline int square (int x) { return x * x; }
```

cout << square (2) << endl; is replaced with cout << ((2) * (2)) << endl; at compile time.

A friend function is a non-member function or ordinary function of a class, which is declared as a friend using the keyword "friend" inside the class.

friend int getAge (Student $); is a friend function which can access the private members of Student class (name, age).

10.3

Page No.
Date :

This program demonstrates an implementation of a basic stack data structure. The size of the stack is defined at run-time and memory is allocated dynamically. The memory the stack takes will not be released until the process arrives at the termination state. But what if the stack was used only for a certain portion of time and then we don't need it for the rest of our process execution. There is a need for a mechanism of dynamically deallocation of memory at run-time. This is where **delete** keyword comes in play.

```
~Stack() {
    delete[] stack;
    cout << "Destructor called " << endl;
}
```

When an object is not used it is removed by the machine and the destructor of that class gets called. In this example, the calling of destructor implies deletion of that dynamically allocated space/memory which the object was holding the pointer to.

10.4

Page No.
Date :

⊖ Function overloading - is a tool that realizes a concept of polymorphism. We could have multiple function with same name, but different number of arguments or different type of argument. More generally two functions are said to be overloading each other if the signature of these function is different but function name is same and return type doesn't matter.

For example :

```
int    sum (int x, int y);
float  sum (float, float);
.
double sum (double, double);

int    sum (int, int, int);
```

All these functions have same name.

Usage

sum (1, 2) ——— sum(int, int)
sum (1, 2, 3) ——— sum(int, int, int)
sum (1.0f, 2.0f) — sum (float, float)
sum (20.5, 10.0) — sum (double, double)

Scanned with OKEN Scanner

10.5

Page No.
Date :

Matrix &Operator + (Matrix &m); is an example of overloaded + operator ~~which~~ function that returns new Matrix which is the resultant sum of this matrix and m matrix.

```
Matrix *s = new Matrix();   ——creates a blank matrix
for (int i=0; i< ROWS; i++)
    for (int j=0; j< COLS; j++)
        s -> matrix[i][j] = this -> matrix[i][j] +
                            m.matrix[i][j];
return *s;
```

Each a_{ij} is sum of ij th elements of this and m matrices.

11.1

Page No.
Date :

```
Matrix & operator = (Matrix &m) {

    for (int i = 0; i < Rows; i++)
        for (int j=0; j < Cols; j++)

            this-> matrix[i][j] = m.matrix[i][j];

    return *this
```

```
Matrix m2 = m1;
```

Assignment operator overloaded function return a reference of this matrix after overwriting the ~~return~~ index values from m1. (in this example).

→ Every index value of m1 is copied to its appropriate location in m2 and when the process is complete m2 has the same entries as m1 does.

Scanned with OKEN Scanner

11.2

Page No.
Date :

Matrix comparison operators compare two matrices ~~and vectors~~ by their elements not reference values. If the matrices have same elements then they are equal otherwise not.

In this program, I have implemented == operator and used the same for != by just putting ! in front of the result that I get from ==.

== and != return boolean value depending upon the match. If both matrices are equal then 1 is returned ~~and~~ or 0 if they are different.

Scanned with OKEN Scanner

11.3

Page No.
Date :

Unary operator have only one operand. some of them are ++, --, !;

Overloading ++ (pre-increment)

```
Student & operator ++ (){
        this->age++
        return *this;
}

Student s = Student("John Doe", 23);

++s; ——→ will increase the age by one

cout << s; ——→ will print the age : 24
```

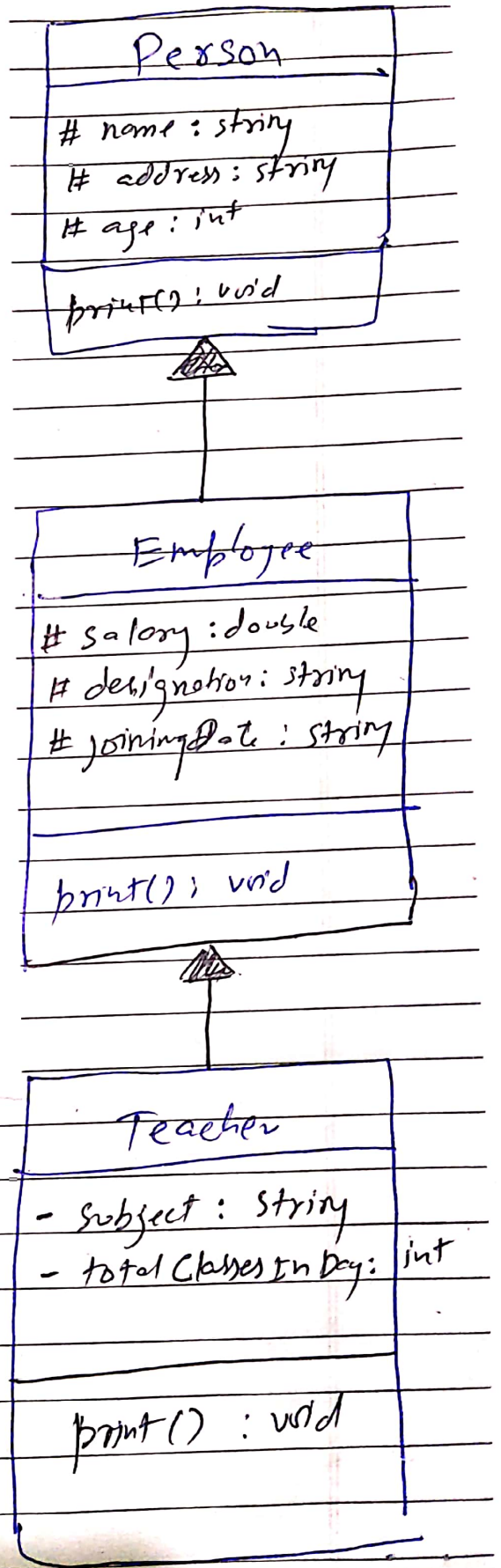
operator overloading is a facility which realizes polymorphism in OOP. Operator overloading is the method by which we can change some specific operators' functions to do different tasks.

Scanned with OKEN Scanner

11-4

Page No.
Date :

```
*** Person Class ***
Name: Shahnawaz
Address: Kulgam
Age: 21

*** Person Class ***
Name: Yawar Abass
Address: Kulgam
Age: 22

*** Person Class ***
Name: Ovais Ahmad Khanday
Address: Anantnag
Age: 23

*** Employee Class ***
Name: Yawar Abass
Address: Kulgam
Age: 22
Salary: 250000
Designation: Software Engineer
Joining Date: 01/01/2024

*** Teacher Class ***
Name: Ovais Ahmad Khanday
Address: Anantnag
Age: 23
Salary: 150000
Designation: Teacher
Joining Date: 01/05/2025
Subject: ToC
Total classes in a day: 2
```



Inheritance

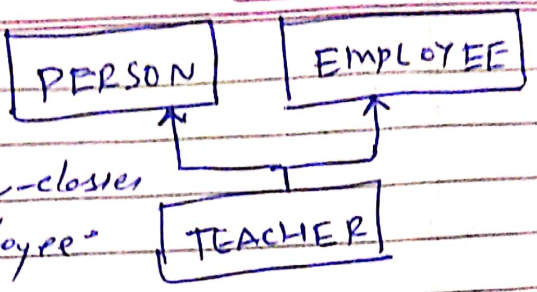
Teacher **is An** Employee **is A** Person

⇒ Teacher objects inherit from Employee and Employee objects inherit from Person

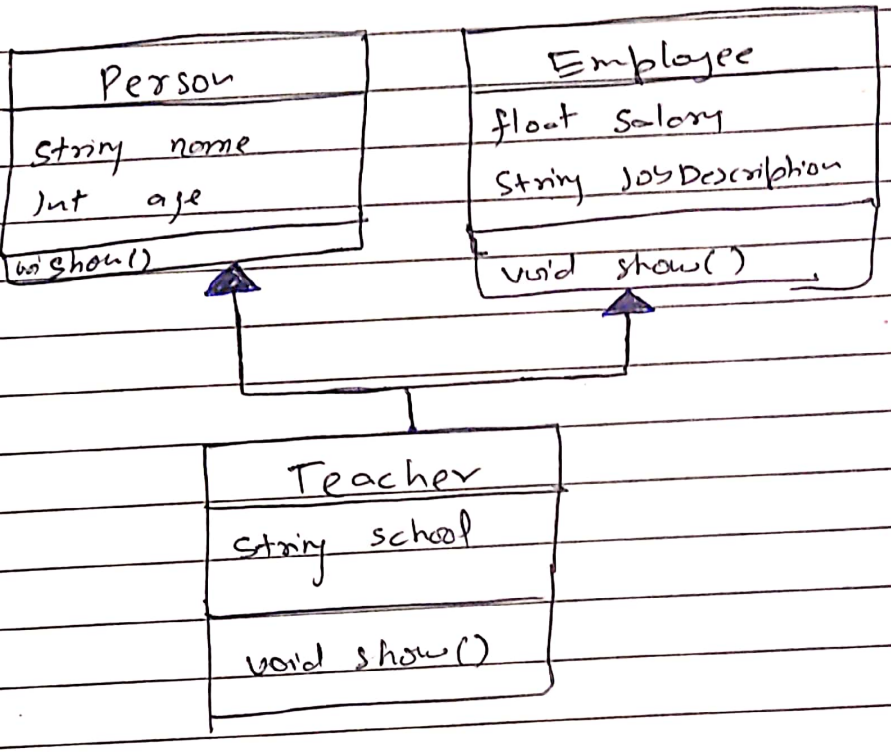
Scanned with OKEN Scanner

12.1

Page No.
Date :



Teacher class : inherit from two super-classes namely Person and Employee. Every teacher object has access to Person and Employee members (not private)



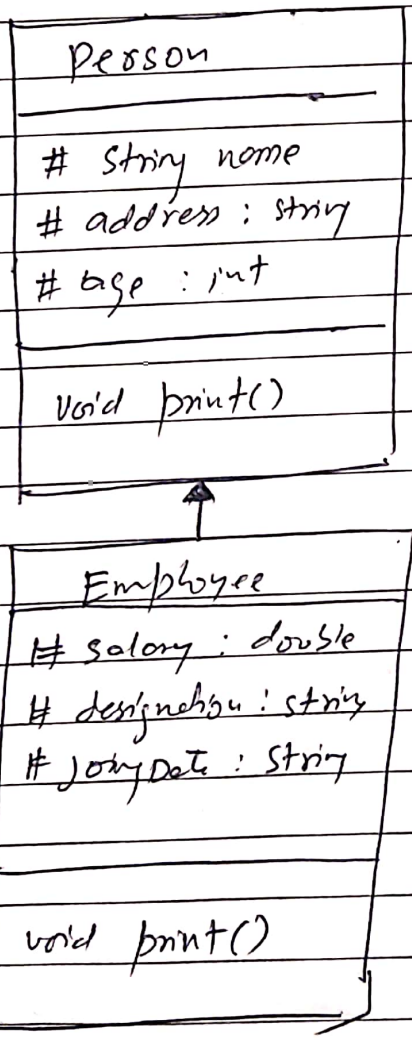

```
Teacher(string name, int age, float salary, string school):
    Person(name, age), Employee(salary, "Teacher") {
        this->school = school;
    }
}
```

→ Sub class constructor Teacher calls superclass constructors Person & Employee ...

12.2

Page No.
Date :

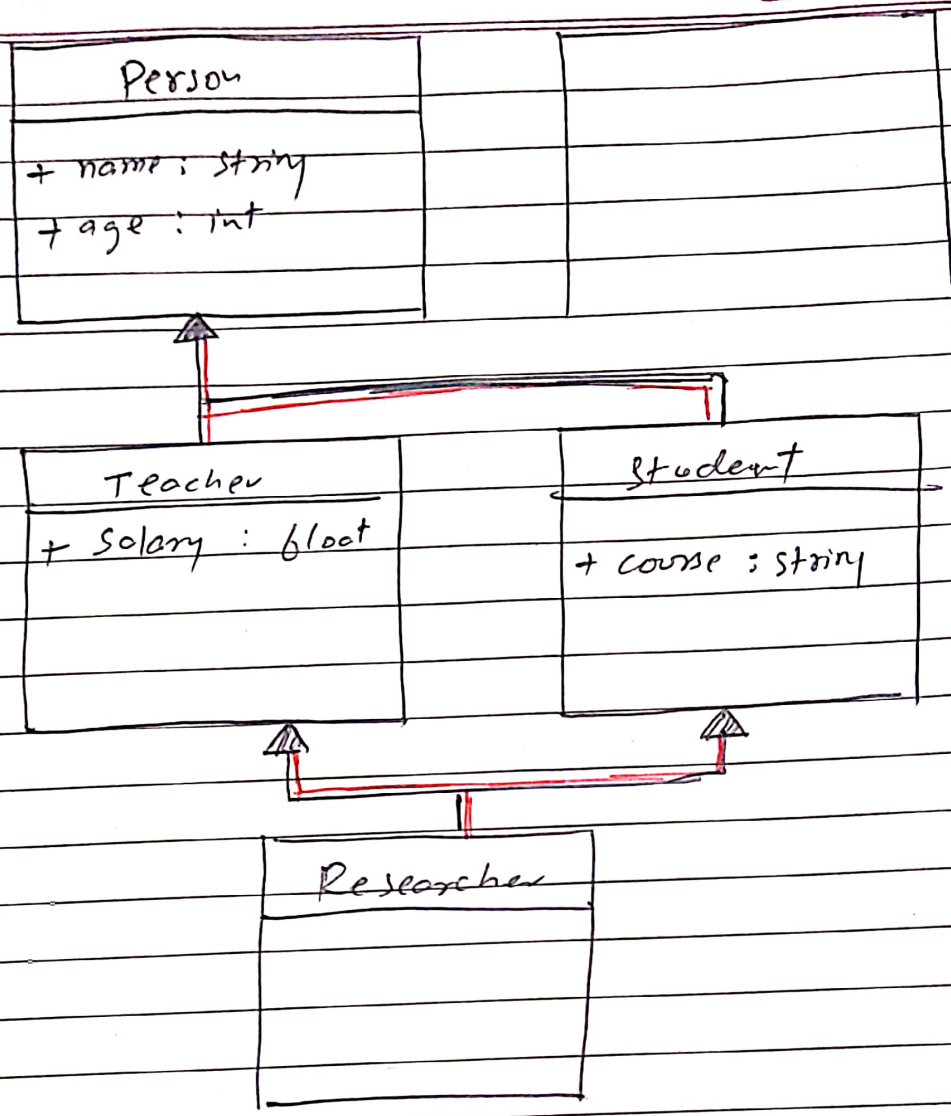
Function Overriding - is a type of polymorphism in which we redefine the member function of a class which it inherited from its base class. The function signature remains same but the working of the function is altered to meet the needs of the derived class. So, when we call the function using its name for the parent object, the parent class function is executed. But when we call the function using the child object, the child class version will be executed.



Same print() function is defined in both classes — Employee::print() function overrides the behavior of Person::print()

12.3

Page No.
Date :



In this example Researcher class inherits from two classes; Teacher and student, which are both inheriting a single classed — Person.

Researcher - class has inherited all attributes of its both superclasses:

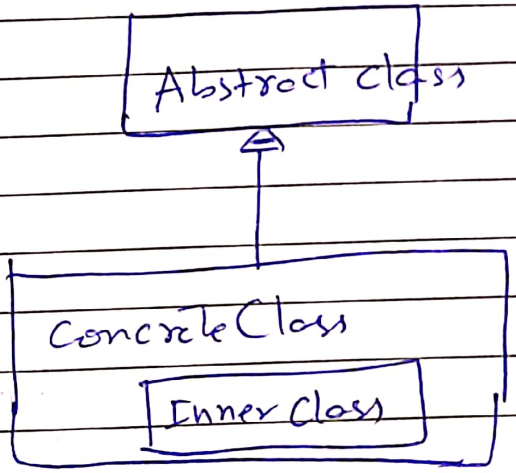
name, age, solary, course.

12.4

Page No.
Date :

A class is said to be an abstract class if it contains at least one pure virtual function —— A pure virtual function has no definition; its definition is deferred to sub-classes inheriting from the base class.

A class defined inside another class is called an Inner class. In this example InnerClass is defined inside Concrete class. The scope of the Inner class is limited to the class in which it has be defined.



Scanned with OKEN Scanner

13.1

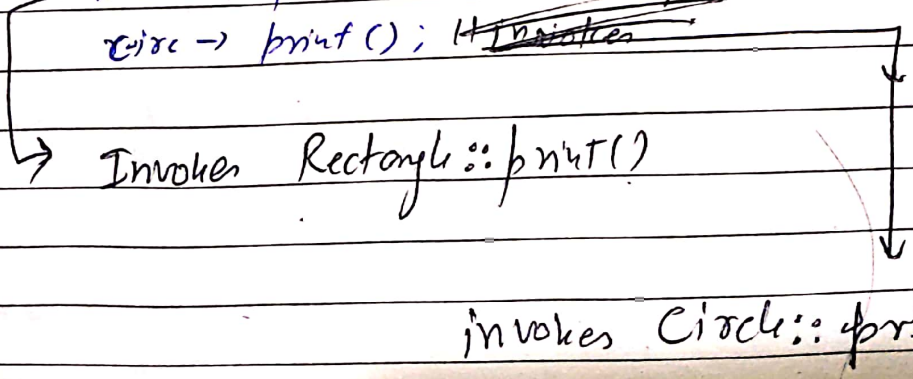
Page No.
Date :

13.1

```
Printing Shape Details
Name: Rectangle
Color: Red
Length: 22
Width: 33
Area: 726
Perimeter: 363

Printing Shape Details
Name: Circle
Color: Blue
Radius: 43
Area: 5805.86
Perimeter: 270.04
```

Virtual functions ~~Eat~~ is a member functions that is declared within a base class and is re-defined (overridden) by a derived class. When you refer to a derived class object using a pointer or a reference to the base class, you can call a virtual function for that object and execute the derived class's version of the method.

```
Shape *rect = new Rectangle("Red", 22, 33);
Shape *circ = new Circle("Blue", 43);

rect -> print(); // invokes Rectangle::print()
circ -> print(); // invokes
```



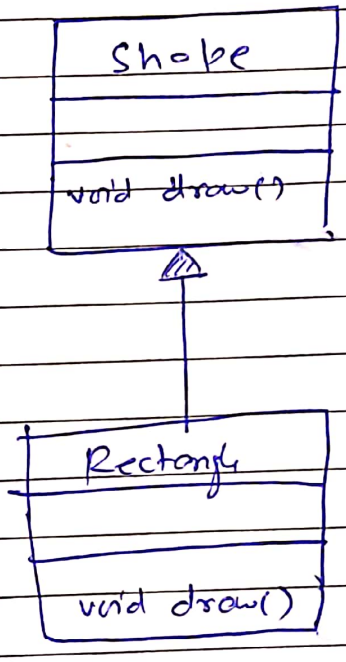
Scanned with OKEN Scanner

13.2

Page No.
Date :

In this example `Shape` is an abstract class, and has a pure virtual function - draw() - which every inheriting class must override or define.

`Rectangle` is one concrete class implementing `Shape`.



Page No.
Date :

```
template <class T>
void swap-(T &x, T &y) {

    T temp = x;
    x = y;
    y = temp;
}
```

This function can be used with predefined or user-defined data-types where assignment operator is defined or overloaded.

```
swap-<int>(ia, ib);
swap-<float>(fa, fb);
.
.
.
```

The same function can work with all those data-types where = operator is defined.

Scanned with OKEN Scanner

13.4

Page No.
Date :

The program in an instance of polymorphism in OOPs. The same Calc class can be used for different types of data. In this program the concept of template classes is being used. Template class is a facility of C++ which realizes polymorphism. A class and its functions are defined for some abstract or general data type (T in most cases).

14.1

Page No.
Date :

Exceptions - technical term of error.

When executing C++ code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things. When an error occurs, C++ will normally stop and generate an error message. The

Exception Handling - in C++ consists of three keywords: try, throw and catch

try - statement allows you to define a block of code to be tested for errors while it is being executed.

throw - keyword throws an exception when a problem is detected, which lets create a custom error.

catch - allows you to define a block of code to be executed, if an error occurs in the try block.

Scanned with OKEN Scanner

14.2

Page No.
Date :

In C++, exception handling is done by throwing an exception in a try block and catching it in the catch block. We generally throw the build-in exceptions provided in the <exception> header but we can also create our own or own custom exception.

To throw a custom exception, we first have to create a custom exception class. This class inherits the std::exception class from <exception> header. We override the what() method of this class to provide a custom error message.

14.3

Page No.
Date :

ALGORITHM

1. Create some Person objects
2. Store these objects in a file
3. Load another file
4. Iterates over all person objects stored in the file at step 2, Increment the salary by 10% and store in file, which is opened at step 3.
5. Prints Person objects from output file to consol.

Scanned with OKEN Scanner