Date. No.

Introduction to C++:

(x): History of C++:

→ It is developed by Bjarne Stroustrup.
→ Started in 1977 at Bell Laboratories as name "C with classes".
- In 1983, ~~C++~~ it was named as C++.

(x) Features of C++:

i) It supports multiple programming paradigms. It is OOP language but also supports POP.

(ii) It is general purpose programming language. It helps us create software in any domain from system software to application software.

(iii): It is a compiled language.

(iv) It is a middle level programming language. ie, it supports both low-level features and high-level features.

Date. No.

(v): It is a portable language ie, it is machine independent but platform dependent.

(vi): It is a powerful / robust language as it supports many built-in datatypes and pre-defined functions.

(vii): It is extensible programming language as we ~~can~~ can add our user-defined functions to the C++ library.

(viii): C++ supports dynamic memory allocation.

(ix): It is case sensitive programming language. ie, we can fetch data from memory using it's address.

(x): It is strongly typed / statically typed language as we must declare variables before using it.

(x) Basic Structure of C++ program:

The C++ program has the following ~~structer~~ sections:

i) Documentation section
ii) Link section
iii) Definition section
iv) Global declaration section.
v) Class definitive section
vi) Main function section
vii) Sub-program function section.

i) Documentation section:
It contains comments of a program and it is optional section.

for single line comment: //
for multiple line comment: /* ... ... */

ii) Link section:
It contains two sections: header and namespace.

*) Header section contains ~~listing of~~ header files to link built-in function.
#include <iostream>

*) Namespace ~~section~~ is declarative region that provides scope to identifiers.
It helps resolve name conflict in program.

iii) Definition Section:
→ It is optional section
→ It contains definitions for symbolic constants.

iv) Global declaration section:
→ It is optional section.
→ It is used to globally declare global variables and functions.

v) Class definition section:
→ It is optional section.
→ It is used to declare classes that contains two subsections:
-> data members -> member function definition.

vi) Main function section:
→ It is complusory section and program execution starts from main function.

vii) Sub-program / User-defined function section:
→ It is optional section.
→ It contains users-defined functions.

Eg:
```
// Author- Kadel
/* Written on
   25th July 2023 */

#include <iostream>

#define PI 3.1415

int r=2;
void area();

class MyClass
{
 public:
 int a;
 void display ()
 {
 std:: cout << "inside class " << std:: endl;
 }
};

void area ()
{
 float area;
 area = PI * r*r
 std:: cout << area << std:: endl;
}

int main ()
{
   MyClass m;
   m.a = 90;
   m.display();
   area ();
   std:: cout << "Hello from main!" << std:: endl;
   std:: cout << m.a << std:: endl;
   return 0;
}
```

→ Output:

```
inside class
12.566
Hello from main!
90
```

*) Note: endl: changes to next line.

X) Reference variable: The variable that refers to the address of another variable.

Date No.

X) Variables:

The name provided to memory location where we store values during program writing is called variables.

Syntax: datatype variablename;

→ Rules:

i) Must begin with alphabet or underscore.
ii) May contain numbes but cant start with it.
iii) Cant have special symbols.
iv) Cant be keywords.

X) Constants:

The fixed values that are used in a program during execution is called constants.

Constants are numeric or string.

Numeric constants are float or integer values.

String constants are single character and strings.

single character is enclosed by ' '
string constant is enclosed by " ".

To declare symbolic constant: #define NAME value.
Using const: const datatype variablename = value;

Date No.

X) Identifiers:

Uses-defined names for variables, functions, classes is called identifiers.

→ They follow the same rules as variables.

→ C++ has no restriction on the length of identifiers but first 31 characters is considered significant.

X) Keywords:

→ Also called reserved words or pre-defined words.

→ They are ~~de~~ words whose meaning are already pre-defined in the C++ compiles library.

→ 95 keywords in C++.

Eg: int, float, const, for, if, inline, etc.

(*)! Note:

cout, cin, main, include, endl, etc are not keywords

They are pre-defined identifiers whose meaning is taken by the C++ compiler. as a () function.

Date. No.

X) Datatypes:

Datatypes is used to define the type of variables stored in a variable.
It is needed to specify how much memory is to be allocated for a variable.

Datatypes are of three categorizes: built-in / primary, desived and user-defined datatypes.

a) Built-in datatypes
- int
- float
- double
- char
- void
- bool
- wchar_t

b) Desived datatype
- array
- pointer
- function
- reference.

c) User-defined datatype.
- class
- structure
- union
- enum
- typedef

data modifiers: signed, unsigned, long, short.

i) int:
- takes 4 bytes
- stores integer values.

Date. No.

(ii): float:
- takes 4 bytes
- stores decimal values upto 7 decimal digits.

(iii) double:
- takes 8 bytes.
- stores upto 15 decimal digits.

*) Note: we can use setprecision(number) to set how many ~~digits~~ of decimal digits taken by float or double.

(iv) char:
- takes 1 byte
- stores single characters.

(*): Note: Typecasting is done by (int) variable name and ASCII value is returned.

(v): bool:
- takes 1 byte.

It returns Boolean value ie, either true or false.

Eg: bool a=true
cout<<a; => 1

bool b=false
cout<<b; => 0.

It is used in conditional cases.
Eg: if ((x>y)==true).

(vi): void:
- It is the special datatype that doesn't return any value.
- It is mostly used for function type.

(vii): wide characters:

Syntax: wchar_t;
It is used to store wide character.
- takes two bytes

Format: wchar_t ch = L'a';

- Mostly used for international value in UNICODE.

x) Input & Output in C++:

For input and output functions, we use iostream header.
The pre-defined objects used are.

cin = standard input ⇒ istream header.

Format: cin (>>) → extraction operator.

cout = standard output ⇒ ostream header

Format: cout (<<) "...."
↳ insertion operator.

Eg:

```
#include<iostream>
int main()
{
  char name[50];
  int age;
  std::cout << "Enter name" << std::endl;
  std:: cin >> name;
  std:: cout << "Enter age" << std:endl;
  std:: cin >> age;

  std:: cout << "The name is " << name << " and age
    is " << age << std:: endl;
  return 0;
}
```

*) Output:

```
Enter name
Ashraya
Enter age
19
The name is Ashraya and age is 19.
```

Date. No.

(X) Operators:

→ Operands: The value on which operator does operations is called operands.

→ Operators: The symbols that perform manipulation on data are called operators.

→ Expression: The sequence of operations and operands giving result is called expression.

Operators are of three types based on number of operands.

i) Unary : one operands
ii) Binary : two operands
iii) Ternary: three operands.

(a): Arithmetic operators:

It is used to perform arithmetic operations on operands.

+ ⇒ add		++	unary increment
- ⇒ subtract		--	unary decrement
* ⇒ multiplication			
/ ⇒ division	X++		--X
% ⇒ modulo (gives remainder)	X--		++X
	postfix		prefix.

Date. No.

(b): Relational operator:

It gives relationship between two operands.

== → equality	!= → not equal to
< → less than	> → greater than
<= → less than equal	>= → greater than equal.

(c) Logical operator:

It gives relationship between two or more conditions and helps to evaluate them.

&& ⇒ logical AND ⇒ gives false if any one false

|| → logical OR ⇒ gives true if any one value true.

! ⇒ logical NOT ⇒ reversed logical condition.

(D): Assignment operator:

Symbol: =

It helps us to assign value to a variable.

(E): Bitwise Operator:

It helps us to perform operation in bit-level.

&	|	^	~	>>	<<
↓	↓	↓	↓	↓	↓
AND	OR	XOR	NOT	right shift	left shift.
returns 1 if both value 1	returns 1 if any value 1	returns 0 if both value same.	reverse 1→0 & 0→1.	a>>units; it shifts given ~~units by~~ number by given units in bit level with number zero.	a<<units;

(F): Conditional operator:

Symbol: ?:

Syntax: (condition) ? true statement: false statement;

(G): Address of:

Symbol: &

It gives address of the given variable.

(F): Cast operator:

It is used for typecasting ie, converting from one datatype to another datatype.

Eg: char ch = '0'
a = (int) ch
cout << a; ⇒ 48

⊛ Operator precedence: The property determining how operators of different types are executed in an expression.

⊛ Operator associativity: The property determining how operators of ~~to~~ same precedence are executed in an expression.

⇒Precedence Order.	Operators	Associativity.
1	() . → ++ (post) -- (post)	L→R
2	++ (pre) -- (pre) + - ! ~ * sizeof()	R→L
3	* / %	L→R
4	+ -	L→R
5	<< >>	L→R
6	< <= >= >	L→R
7	== !=	L→R
8	&	L→R
9	^	L→R
10	|	L→R
11	&&	L→R
12	||	L→R
13	?:	L→R
14	= += -=	R→L.
15	,	L→R