Date No.

X) Differences between Reference Variable and Pointers

References	Pointers
Variable cannot be reassigned in reference.	- Variables can be reassigned in pointers.
Shares same address as original variable	- ~~Star~~ has their own memory access.
refers to another variable.	- stores address of another variable.
It doesn't have NULL value.	- NULL value can be assigned.
This variable is referenced by method pass by value.	- The variable does work by method pass by reference.

X) Differences between Inline and Normal Function.

Inline	Normal.
It expands the code inline when invoked.	- It provides modularity to the program.
It is used when small functions called very often.	- It is used to improve reuseability of code making it maintainable.
Requires 'inline' keyword declaration.	- No keyword declaration.
Execution is generally faster than normal functions.	- Execution is generally slower than inline functions.
Compiler places copy of the code of the function at compile time.	- Compiler doesn't paste code inline.
Funtions present inside class are implicitly inline.	- Functions outside classes are normally normal functions.
Use of too many inline function increases the size of executable file.	- Use of normal functions doesn't affect the size of executable file.

(X): Namespace:

A namespace is a declarative region that provides a scope to the identifiers inside it.

It is used to organize codes into logical groups and to prevent name collisions that can occur if codebase includes multiple libraries.

*) Syntax:

```
namespace namespace_name {
   statements;
}
```

*) Points to remember:

i) Namespace declarations only occurs at global scope.

ii) Namespace declarations can be nested.

iii) No need of semi-colon after closing brace.

iv) Namespace definition can be split over several units.

X) Strings:

Strings are objects of class std:: string in C++ used to represents sequence of characters.

It has header file <string>.

*) Defining string: std:: string str_name = "string";

x) Input cin >> str; => takes string except from spaces

getline (cin, str); => takes whole string including space.

* Output: cout << str;

→ Indexing of string is the same as indexing of array

⇒ Difference between string and character array.

string	character array.
→ string is objects of class std:: string	→ array of character datatype
→ declaration: string str_name;	→ declaration: char array_name [size];
→ DMA i.e, no pre-allocated memory	→ SMA i.e, pre-allocated memory.
→ Have many inbuilt functions.	→ No inbuilt functions.
→ slower implementation	→ faster implementation

Date. No.

⇒ string functions:

a) reverse(): It reverses string from starting point to end point.

Syntax: reverse (ptr-start, ptr-end);
Eg: string str = "abcd";
reverse (str.begin(), str.end());
points to beginning of string; points to ending of string.

b) substr() ⇒ used to find substring of given string

Syntax: str_name.substr (position, length);
Eg: string str = "Hello";
~~sub~~ str.substr (0,3); ⇒ Hel.

c) + : used to concatenate strings.

Eg:
string s1 = "college";
string s2 = "day";
cout << s1+s2 << endl; ⇒ collegeday

if s1 = s1+s2 ⇒ use of extra memory
s1 += s2 ⇒ uses extra space.

Date. No.

(d): pushback(): It inserts characters at the end of string.

Syntax: str_name.pushback (character);
Eg: string str = "abcd";
char ch = 'e';
str.pushback (ch); ⇒ abcde.

(e): size()/length():
It gives the length of string.

Syntax: str_name.size();
str_name.length();

(f) strlen (~~str~~) ⇒ gives length of characters array.

(g): to_string() ⇒ converts integer value to string.
Eg:
int num = 432;
to_string (num); ⇒ "432"

(h): find(): returns index of occurence of substring.

Syntax: str_name.find (substr, position, length);
~~or~~ position and length are optional.

(i): replace(): It replaces part of string with another string.

Syntax: str_name.replace(position, length, string);

(j): insert(): It inserts characters in strings at specified position.

Syntax: str_name.insert(position, string);

(k): strncmp(): It is used to compare strings. length. It is in <cstring> header.

Syntax: (str1, str2, length);
It is done using character pointer.

```
char* str1 = "....";
char* str2 = "....";
```

- It returns 0 if str1 = str2.
- It returns less than 0 if str1 < str2.
- It returns to more than 0 if str1 > str2.

x) Comparison of C and C++:

C	C++
C is subset of C++.	- C++ is superset of C.
C supports procedural programming paradigm.	- C++ supports both procedural and object oriented programming paradigm.
C doesn't supports OOP.	- C++ supports OOP.
In C, data and functions are separates and free entities.	- In C++, data and functions are encapsulated together in form of object.
In C, data are free entities and can be manipulated by outside code.	- In C++, encapsulation hides data to ensure data security.
- It is function driven language	- It is object driven language.