Q1. What is template, Explain Function and class template in detail.

Ans. To perform identical operations for each type of data compactly and conveniently, the function templates are used.

- Syntax of function templates -

  template < class name of data types >
  name _of_ datatype funct _name (name _of_ datatype id1, ...
  name _of _data_ type id2)

- eg:

```
template   <class T>
T min (T a, T b)
{
        if (a < b).
        return a;
        else
        return b;
}
int main ()
{
        cout << min (10,20) << endl;
        cout << min ('p', 't') << endl;
}
```

O/P:
10
P

- Using class template we can write a class whose members use template parameters as types.
- Syntax of class template :-

  template < class Type >
  class class_name
  {
        ....body of class.
  }

```
eg: template <class T>
class  Compare
    {
            T  a,b;
            public :
            Compare (T first, T second)
            {    a = first;
                 b = second;
            }
            T  max ();
    };
    template <class T>
    T  Compare <T> :: max ()
    {
            T  val;
            if (a > b)
                val = a;
            else
                val = b;
            return val;
    }
    int  main ()
    {
            Compare <int> obj 1 (100, 60) ; cout << obj 1.max ();
            Compare <char> obj 2 ('p', 't'); cout << obj 2. max();
            return  0;
    }
    O/P:
    100
    t
```

Q2. Explain the power of template.

Ans. The power of templates :-

1. Templates are useful for code reusability.

2. The generic functions and generic classes are useful tool for creating the generalized code that can handle any datatype elements.

3. The Standard Template Library (STL) is built upon the concept of templates, in which the commonly used library classes or functions are written as templates.

4. It helps in generating high performance object code.