TUTORIAL - 15

1)
a) function
b) virtual
c) base class
d) 0
e) derived class
f) base
g) abstract
h) pure virtual function
i) override
j) inherited

2) Output : In Derived
In Derived

The pointer has been class type but points to derived class object.

3) The virtual mechanism works only when we have a base class pointer to a derived class object. In C++, the constructor cannot be virtual, because when a constructor of a class is executed there is no virtual table in the memory, means no virtual pointer defined yet. So, the constructor should always be non-virtual

the code will not compile.

4) Virtual functions are invoked when you have a pointer/reference to an instance of a class. Static functions aren't tied to a particular instance, they're tried to a class. so no static functions can't be virtual this code won't compile.

5)

```
#include <iostream>
using namespace std;
class Animal
{
public:
 virtual void display()
  {
  cout <<" I am a Animal"<< end1;
 }
};
class Mammal: public Animal
{
Public:
  virtual void display()
  {
  cout << " I am a Mammal " << end1;
  }
};
class Reptile :public Animal
 {
Public:
   virtual void display()
{
cout <<" I am a Reptile" << end1;
```

```
};
int main()
{
Animal *a = new Reptile;
Animal *b = new Mammal;
Animal *c = new Animal;
Reptile *d = new Reptile;
Mammal *e = new Mammal;

  a->display();
  b->display();
  c->display();
  d->display();
  e->display();
}
```

Output:

```
I am a Reptile
I am a Mammal
I am a Animal
I am a Reptile
I am a Mammal.
```

6)
```
# include <iostream>
  using namespace std;
template < typename T>

  void sorting (T a[], int n)
  {
  int i, j ;
  T t;
  cout <<"\n Before sorting :" <<endl;
for (i=0 ; i<n; i++)
  {
  cout << a [i] <<" ";
}
for (i=0; i<n-1; i++)
  {for (j=0; j<n-i-1 ; j++)
  {
  if (a[j] > a [j+1])
  {
  t= a [j];
  a[j] = a[ j+1];
    a[j+1) = t ;
    }
  }
}
cout <<"\n After sorting:" <<endl;
      for (i=0; i<n; i++)
      {
      cout <<a [i]<<" ";
      }
```

```
}
int main()
 {
int a[] = {5,2,4,1,3};
float b[] = {2.2, 4.8, 1.6, 3.7, 5.2};
  char c[] = {'a', 'e', 'b', 'd', 'c'};
  Sorting (a,5);
  Sorting (b,5);
  sorting (c,5);
 }
```

Output:

Before sorting:
5 2 4 1 3
After sorting:
1 2 3 4 5

Before sorting:
2.2 4.8 1.6 3.7 5.2

After sorting:
1.6 2.2 3.7 4.8 5.2

Before sorting:
a e b d c

After sorting:
a b c d e

7.) 1 2 3 4 5

8) functions overloading is used when multiple functions do similar operations, templates are used when multiple functions do identical operations. templates provide an advantage when you want to perform the same action on types that can be different.

9)
```
#include <iostream>
using namespace std;
template <typename T>
class vector
{
T* ptr;
int size, capacity;
Public : vector();
void push_back (T val);
  void print();

};
template <typename T>
 vector <T>:: vector()
{
ptr = new T[10];
 Capacity = 10;
   size = 0;
}
template <typename T>
 void vector <T>:: push_back (T val)
 { int i;
    if (capacity <= size)
    {
```

```
capacity *= 2;
T * p = ptr;
ptr = new T [capacity];
for (i=0; i<size; i++)
  {
  ptr [i] = p [i];
  }
}

  ptr [size] = val;
  size ++;
}
template <typename T>
void vector <T>:: prix ()
  {
  int i;
  for (i=0; i< size; i++)
  {
cout << "  " << "(ptr+i);
  }
  cout << end 1;
}
int main ()
{
    int  i, v, c = 1;
  vector <int> a [5];
  while (c == 1)
  { cout <<" enter vector to add element to "<<end1;
  cin >>i;
  cout <<" enter value to push " << end 1;
```

```
cin >>v;
cout << " Enter 1 to add more elements" << endl;
cin >>c;
   a[i-1].push_back (v); }

   for (i=0; i<5; i++)
    {
   a[i]. print();
     }
}
```

Output -

```
ANISHAs - Mac Book - Air : - anishasaxena $ / users / anissha
  Saxena / Desktop / array;vector
 enter vector to add element to
 1
enter palue to push
 12
Enter 1 to add more elements
 1
Enter vector to add element to
 3
enter value to push
  33
Enter 1 to add more elements
 1
 enter vector to add element to
 5
enter value to push
 14
 Enter 1 to add more elements
 1
 Enter vector to add element to
```

2
Enter value to push
15
Enter 1 to add more elements
1
enter vector to add element to
4
enter value to push
16
Enter 1 to add more elements
1
enter vector to add element to
3
enter value to push
17
Enter 1 to add more elements
1
Enter vector to add element to
3
enter value to push
18
Enter 1 to add more elements.
1
enter vector to add ~~to~~ elements to
2
enter value to push
19
Enter 1 to add more elements
1
enter vector to add element to
4
enter value to push
20
Enter 1 to add more elements
0
12

15
13 19
16 12 18
14 20

10)
```
#include <iostream>
using namespace std;
template <typename T>
class vector
 {
 T * ptr;
int size, capacity;
Public : vector();
void push_back (T val);
void print();
void modify (int p, T v);
 void scalar (T s);
};
template <typename T>
vector <T> :: vector()
   {
   ptr = new T[10];
    capacity = 10;
     size = 0;
   }
template <typename T>
void vector <T> :: push_back (T val)
 { int i;
    if ( capacity <= size)
 {
 capacity *= 2;
```

```
    T * p = ptr;
    Ptr = new T[capacity];
    for (i=0; i<size; i++)
      {
       ptr[i] = p[i];
      }
}     p.
      ptr[size] = val;
      size++;

  }
  template <typename T>
  void vector <T> :: print ()
   {
   int i;
   for (i=0; i<size ; i++)
     {
   cout << "    " << * ( ptr+i );
   }
   cout << end l;
 }
template < typename T >
void vector <T> :: modify (int P, T V)
     {
    * (ptr + P) = V;
  }
  template <typename T>
  void vector <T> :: scalar (T S)
```

```
{
int i;
for(i=0; i<size; i++)
{
*(ptr+i)*=s;
}
}
int main()
{
int i,v,c=1;
vector<int>a;
while(c==1)
{
cout<<"Enter value to push"<<endl;
cin>>v;
cout<<"Enter 1 to add more elements "<<endl;
cin>>c;
a.push_back(v);}
cout<<"original vector:"<<endl;
a.print();
cout<<"Enter position to modify"<<endl;
cin>>i;
cout<<"enter value to push"<<endl;
cin>>v;
a.modify(i,v);
```

```
cout <<" vector After modify:" << end l;
a. print ();
cout <<" Enter scalar " << end l;
cin >> v;
 a. scalar (v);
 cout <<" vector After scalar multiplication:" <<end l;
 a. print ();
 }
```

Output -

```
Enter 1 to add more elements
1
Enter value to push
14
Enter 1 to add more elements
1
enter value to push
15
Enter 1 to add more elements
1
enter value to push
16
Enter 1 to add more elements
1
Enter value to push
17
Enter 1 to add more elements
1
enter value to push
18
```

```
Enter 1 to add more elements
1
enter value to push
12
Enter 1 to add more elements
1
enter value to push
20
Enter 1 to add more elements
1
enter value to push
21
Enter 1 to add more elements
1
enter value to push
22
Enter 1 to add more elements
1
enter value to push
23
Enter 1 to add more elements
1
enter value to push
24
Enter 1 to add more elements
1
enter value to push
25
Enter 1 to add more elements
0
original vector:
12 13 14 15 16 17 18 19 20 21 22 23 24 25
```

Enter position to modify
0
Enter value to push
26
Vector After modify:
12 13 14 15 16 17 18 19 20 21 22 23 24 25
Enter scalar
5
Vector After scalar multiplication!
60 65 70 75 80 85 90 95 100 105 110 115 120 125