Experiment 5: Program to understand different types of constructors and destructor.

Constructor: In C++, constructor is a special method which is invoked automatically at the time of object creation. It is used to initialize the data members of new object generally. The constructor in C++ has the same name as class or structure.

Types of Constructors:

1. Default Constructor
2. Parametrized Constructor
3. Copy Constructor

Default Constructor: A constructor which has no argument is known as default constructor. It is invoked at the time of creating object.

```
#include <iostream>
using namespace std;
class Employee
 {
  public:
      Employee()
      {
         cout<<"Default Constructor Invoked"<<endl;
      }
};
int main(void)
{
   Employee e1; //creating an object of Employee
   Employee e2;
   return 0;
}
```

Output: Default constructor invoked
Default constructor invoked

Parameterized Constructor: A constructor which has parameters is called parameterized constructor. It is used to provide different values to distinct objects.

```
#include <iostream>
using namespace std;
class Employee {
  public:
     int id;//data member (also instance variable)
     string name;//data member(also instance variable)
     float salary;
     Employee(int i, string n, float s)
      {
         id = i;
         name = n;
```

```
        salary = s;
      }
     void display()
      {
        cout<<id<<"  "<<name<<"  "<<salary<<endl;
      }
};
int main(void) {
   Employee e1 =Employee(101, "Sonoo", 890000); //creating an object of Employee
   Employee e2=Employee(102, "Nakul", 59000);
   e1.display();
   e2.display();
   return 0;
}
```

Output: 101 Sonoo 890000
102 Nakul 59000

Destructor: A destructor works opposite to constructor; it destructs the objects of classes. It can be defined only once in a class. Like constructors, it is invoked automatically.

A destructor is defined like constructor. It must have same name as class. But it is prefixed with a tilde sign (~).

```
#include <iostream>
using namespace std;
class Employee
 {
   public:
      Employee()
      {
        cout<<"Constructor Invoked"<<endl;
      }
      ~Employee()
      {
        cout<<"Destructor Invoked"<<endl;
      }
};
int main(void)
{
   Employee e1; //creating an object of Employee
   Employee e2; //creating an object of Employee
   return 0;
}
```

Output:

```
Constructor Invoked
Constructor Invoked
Destructor Invoked
Destructor Invoked
```