CHAPTER-3
Function Overloading
SHORT ANSWER QUESTIONS

1.	**How does the compiler interpret more than one definitions having same name? What steps does it follow to distinguish these?**
Ans.	The compiler will follow the following steps to interpret more than one definitions having same name: (i) if the signatures of subsequent functions match the previous function's, then the second is treated as a re-declaration of the first. (ii) if the signature of the two functions match exactly but the return type differ, the second declaration is treated as an erroneous re-declaration of the first and is flagged at compile time as an error. (iii) if the signature of the two functions differ in either the number or type of their arguments, the two functions are considered to be overloaded.
2.	**Discuss how the best match is found when a call to an overloaded function is encountered? Give example(s) to support your answer.**

Ans.	In order to find the best possible match, the compiler follows the following steps:
	1. Search for an exact match is performed. If an exact match is found, the function is invoked. For example,
	2. If an exact match is not found, a match trough promotion is searched for. Promotion means conversion of integer types char, short, enumeration and int into int or unsigned int and conversion of float into double.
	3. If first two steps fail then a match through application of C++ standard conversion rules is searched for.
	4. If all the above mentioned steps fail, a match through application of user-defined conversions and built-in conversion is searched for.
	For example,
	`void afunc(int);`
	`void afunc(char);`
	`void afunc(double);`
	`afunc(471); //match through standard conversion. Matches afunc(int)`

3.	**Discuss the benefits of constructor overloading. Can other member function of a class be also overloaded? Can a destructor be overloaded? What is your opinion?**
Ans.	Constructor overloading are used to increase the flexibility of a class by having more number of constructors for a single class. By this we can initialize objects more than one way. Yes, Other member function of a class can also be overloaded. A destructor cannot be overloaded.

4.	**Write the output of the following C++ code. Also, write the name of feature of Object Oriented Programming used in the following program jointly illustrated by the functions [I] to [IV]:**
	
	`#include<iostream.h>`
	`void Line()  //Fuction [I]`
	`{  for(int L=1;L<=80;L++) cout<<"-";`
	`  cout<<endl;`
	`}`
	`void Line(int N)  //Fuction [II]`
	`{  for(int L=1;L<=N;L++) cout<<"*";`
	`  cout<<endl;`
	`}`
	`void Line(char C,A,int N)  //Fuction [III]`
	`{  for(int L=1;L<=N;L++) cout<<C;`
	`  cout<<endl;`
	`}`
	`void Line(int M,int N)  //Fuction [IV]`
	`{  for(int L=1;L<=N;L++) cout<<M*L;`
	`  cout<<endl;`
	`}`
	`void main()`

```
{     int A=9,B=4,C=3;
      char K='#';
      Line(K,B);
      Line(A,C);
}
```

Ans.

Output:

# # # #

9 18 27

The name of feature of Object Oriented Programming used in the above program jointly illustrated by the functions [I] to [IV] is known as '*function overloading*'.

5. **Here are some desired effects. Indicate whether each can be accomplished with default arguments, with function overloading, with both, or which neither. Provide appropriate prototypes.**

(i) repeat (10, '-') displays the indicated character ('-' in this case) given number of times (10 here). While repeat() displays '*' character 12 times. Also repeat('#') displays the given character ('#' here) 12 times and repeat (7) displays '*' given no of times (7 here).

(ii) average (4,7) returns int average of two int arguments, while average (4.0, 7.0) returns the double average of two double values.

(iii) mass (density, volume) returns the mass of an object having a density of density and a volume of volume, while mass (density) returns the mass having a density of *density* and a volume of 1.0 cubic meters. All quantities are type double.

(iv) average (4,7) returns an int average of the two int arguments when called is one file, and it returns a double average of the two int arguments when called in a second file in the same program.

(v) handle (a-character) returns the reversed case of the passed character or prints it twice depending upon whether you assign the return value to it or not.

Ans. (i) It can be accomplished with function overloading.

```
void repeat(int n, char c);
void repeat();
void repeat(char c);
void repeat(int n);
```

(ii) It can be accomplished with function overloading.

```
int average(int a,int b);
double average(double a,double b);
```

(iii) It can be accomplished with default argument.

```
double mass (density d, volume v);
double mass(density d, volume v=1.0);
```

(iv) It can be accomplished with function overloading.

```
int average(int a,int b);
double average(int c,int d);
```

(v) It can be accomplished with function overloading.

```
char handle(char a);
void handle(char a);
```

6. **Write function definitions for question 7 on page 119.**

Ans.

```
int themax(int a)
{
```

```
            return a;
      }
      int themax(int a,int b)
      {
            if(a>b)
                return a;
            else
                return b;
      }
      int themax(int a[])
```

7. **Write function definitions for question 8 on page 119.**
Remember that cube's volume is side^3
Cylinder's volume is $\pi r^2 h$
Rectangular box's volume is length x breadth x height.

Ans.

```
      float volume(float side)
      {
            return side*side*side;
      }
      float volume(float radius, float height)
      {
            return 3.14*radius*radius*height;
      }
      float volume(float length, float breadth, float height);
      {
            return length*breadth*height;
      }
```

8. **Write definitions for two versions of an overloaded function. This function's 1st version sum() takes an argument, int array, and returns the sum of all the elements of the passed array. The 2nd version of sum() takes two arguments, an int array and a character ('E' or 'O'). If the passed character is 'E', it returns the sum of even elements of the passed array and is the passed character is 'O', it returns the sum of odd elements. In case of any other character, it returns 0 (zero).**

Ans.

```
int sum(int a[])
{
      int n,sum=0;
      cout<<"Enter n:";
      cin>>n;
      for(int i=0;i<n;i++)
      {
            sum=sum+a[i];
      }
      return sum;
}
//
int sum(int a[],char c)
{
      int even=0,odd=0;
      switch(c)
      {
            case 'E':
                  for(int i=0;i<5;i++)
                  {
                        if(a[i]%2==0)
                        {
                              even=even+a[i];
                        }
```

```
                }
                return even;
            case 'O':
                for(int j=0;j<5;j++)
                {
                    if(a[j]%2!=0)
                    {
                        odd=odd+a[j];
                    }
                }
                return odd;
        }
}
```

9. **Compare the usefulness of default argument and function overloading, supporting your answer with appropriate examples.**

Ans. Default values can be provided the function prototype itself and the function may be called even if an argument is missing. But there is one limitation with it, if you want to default a middle argument, then all the argument on its right must also be defaulted. For instance, consider the following function prototype: `float amount(float p, int time=2, float rate=0.08);`

```
    cout<<amount(2000,0.13);
```

Here, C++ will take 0.13 to be the argument value for time and hence invoke amount() with values 2000, 0(0.13 converted to int) and 0.08

Function overloading can handle all possible argument combinations. It overcomes the limitation of default argument but also compiler is saved from the trouble of testing the default value.

Example:

```
float area(float a)
{
    return a*a;
}
float area(float a,float b)
{
    retur a*b;
}
```

10. **Raising a number n to a power p is the same as multiplying n by itself p times. Write as overloaded function power() having two versions for it. The first version takes double n and int p and returns a double value. Another version takes int n and int p returning int value. Use a default value of 2 for p in case p is omitted in the function call.**

Ans.

```
double power(double n,int p=2)
{
    double res=pow(n,p);
    return res;
}
int power(int n,int p=2)
{
    int res=pow(n,p);
    return res;
}
```

LONG ANSWER QUESTIONS

1. **Given below are incomplete definitions of two classes:**

```
class Item {
        int itemno;
        char description[21];
        int QOH;  // Quantity-On-Hand
```

```
        int ROL;  // Reorder-Level
        int ROQ;  // Reorder-Quantity
      public:
        :
        // Overloaded Constructor Definition
        void purchase(int n)
        {
            :    // adds n to QOH
        }
        void sale(int n)
        {
            :    // subtracting n from QOH
        }
        void display(void)
        {
            // display item details
        }
};
class store{
        Item itemlist[50];
              //List of 50 items
        :
      public:
        :          //constructor(s)
         new()
         {
          :
              //To add new item into itemlist
         }
              delete()
         {
           :    //To delete new item into itemlist
         }
         order(itemno, qty)
         {
            :  //To order qty pieces for itemno
         }
         order()
         {
          :    //check which item's QOH is
          :    // below their ROL, then order for
          :    // ROQ pieces for all those itemnos
         }
         bill()
         {
          :    //Ask itemno, qtysold and
          :    // prepare the bill. Also update
          :    // Item's information by
          :    // calling purchase() of Item.
         }
        };
```

Using above information, write a complete C++ program that lets you sell, purchase or order a specific item or items.

Ans. **Code snippet of the above problem is given below use it and complete the program.**