Page No.
Date

Experiment 7.

a. WAP using function overloading to calculate the area of a laboratory & area of classroom.

→

```
#include <iostream>
using namespace std;
class Area {
Public:
   int calculate (int length, int width) {
      return length * length;
   }
int calculate (int side) {
      return side * side;
   }
};
int main () {
   Area a;

cout << " Area of labotory (rectangle 10x20): " << a.
        calculate (10,20) << endl;

cout << " Area of classroom (square 15x15): " << a.
        calculate(15) << endl;

      return 0;
}
```

- Output:

```
Area of laboratory (Rectangle 10x20) : 200
Area of classroom (square 15x15)  : 225
```

b) WAP using function overloading to calculate sum of 5 float values & sum of 10 integer values.

→

```
#include <iostream>
using namespace std;
class calculator {
 int num;
 public:
 calculator (int n=0) { num = n };
 Calculator operator + ()
 {
  return calculator (-num);
 }
 Void showNum () {

  cout << "value:" << num << endl;
 }
 float add (float a, b, c, d, e) {
     return a+b+c+d+e;
 }
int add (int a, b, c, d, e, f, g, h, i, j ) {
     return a+b+c+d+e+f+g+h+i+j ;
   }
};
int main () {
   calculator c1 (10);
   c1. showNum();

   calculator c2 = + c1;
    c2. showNum();

   cout << " sum of 5 floats:" << c1. add(1.1, 2.2, 3.3, 4.4, 5.5) << endl;
   cout << " sum of 10 int :" << c1. add (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) << endl;
```

```
return 0;
}
```

- Output:

```
Value              :  10
value              :  -10
sum of 5 floats :  16.5
sum of 10 int   :  55
```

c) WAP to implement unary -operator when used with the object so that the numeric data members of class is negated.

→

```
# include <iostream>
using namespace std;
class Number {
    int value;
  public:
   Number (int v) {
   value = v;
}
void display () {
 cout <<" value : " <<value <<endl;
 } }
void display ( string msg) {
 cout << msg <<value <<endl;
}
void operator - () {
    value = -value ;
 }
};
```

```
int main()
{
  number n(10);

  n.display();
  n.display("current value:");
  -n;
  n.display("After negation:");
    return 0;
}
```

- Output:

```
value          : 10
current value  : 10
After negation : -10
```

d) WAP to implement unary ++ operator (for pre increment & post increment) when used with the object so that the numeric data members of the class is incremented.

→

```
#include <iostream>
using namespace std;
class Demo{
   int num;
   public:
     Demo(int n=0){
     num=n;
   }
  void show(){
  cout<<"Number:"<<num<<endl;
  }
```

```
void show ( string msg) {
 cout << msg << num << endl ;
 }
Number operator ++ () {
     ++ num;
     return *this;
  }
Number operator ++(int) {
   number temp= *this;
    num++;
     return temp;
  }
};
int main () {
  number n1(5);

  n1.show();
  n1.show("value:");
  ++ n1;
  n1.show("After Pre-increment:");
  n1++;
  n1.show("After Post-increamant:");
}
```

- output:

```
Number          : 5
value           : 5
After Pre-increment : 6
After Post-increment : 7
```

17/10

Page No.
Date

Experiment - 8

* WAP to demonstrate compile time & Run time Polymorphism (operator overloading binary).

a) WAP to overload the '+' operator so that two strings can be concatenated.

→
```
#include <iostream>
#include <string>
using namespace std;
class Mystring {
    string str;
Public:
    Mystring (string s) : str(s) {
    }
Mystring operator + (const Mystring &s) {
    return mystring (string + s.string);
}
Void display() {
 cout << str << endl;
   }
};
int main() {
 Mystring s1 ("Hello") , s2("world!");
 Mystring s3 = s1 + s2;
 s3. display();
}.
```

■ Output :

Hello , World !

b) WAP to create a base class Login having data members as name & password, declare accept() function virtual. Derive Email login & Membershiplogin classes from login. Display Email login details & membership login details of emp.

→
```
#include <iostream>
using namespace std;
class Number {
  int value;
 Public:
  Number (int v=0) {
    value = v;
  }
  Number operator +(Number n) {
   return Number( value +n. value);
  }
  void display() {
    cout << "value:" << value << endl;
  }
};
class login {
 Protected: string name, Password;
   Public:
      virtual void accept() {
        cin >> name >> Password;
      }
     virtual void accept() {
     }
};
Class Emaillogin: Public login {
      string email;
 Public:
```

```
Void accept() {
    login :: accept();
    cin >> email;
}
Void display() {
    cout << " email login: " << name << " " << Password << " " << email
        << endl;
}
};
class membershiplogin : Public login {
    string memId;
Public:
    Void accept() {
        login:: accept();
        cin >> mem ID;
    }
    void display() {
    cout << " membership login: " << name << " " << Password << " "
                                    memId << endl;
    }
};
int main() {
    number n1(10), n2(20), n3;
    n3 = n1 + n2;
    n3.display();
    login *t;
}
```

17/10

■ output:

value : 30

Email login : Alice Pass123 alice@gmail.com

membership login : bob Pass456 M123.

a) WAP

→ #in
#in
usin
int n
if
of

if (
co
}
strin
whi
}
cou
in
ou
}

■ output

Experiment - 9

Page No.
Date

a) WAP to copy contents of one file into another.

→

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
ifstream infile("first.txt");
ofstream outfile("second.txt");

if(!infile){
  cout<<"first.txt not found!"<<endl;
  return 1;
}
string line;
while(getline(infile,line)){
   outfile<<line<<endl;
}
 cout<<"context copied succesfully!"<<endl;
 infile.close();
 outfile.close();
}
```

■ output:

contents copied succesfully!

b) WAP to count digits & spaces using file handling.

→
```
#include <iostream>
#include <fstream>
using namespace std;
int main() {
  ifstream file("input.txt");
  char ch;
  int digits=0; spaces=0;
  while(file>>noskipws>>ch) {
    if(ch>='0' && ch<='9') digits++;
    if(ch==' ') spaces++;
  }
  cout<<"Digits:"<<digits<<"\nSpaces:"<<spaces<<endl;
  file.close();
}
```

■ output:

digits=0
spaces=0

c) WAP to count words using file handling.

→
```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
ifstream file("input.txt");
string word;
int count=0;
```

```
while (file >> word)
    count++;
cout << " total words : " << count << endl;
file.close();
}
```

- Output:

Total Words : 0

d) WAP to count occurence of a given word using file handling.

→

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
int main() {
  ifstream file("sample.txt");
  string word, target;
  int count = 0;
  cout << " enter word to count: ";
  cin >> target;
  while (file >> word) {
    if (word == target) {
      count++;
    }
  cout << " the word " << target << " occurs " << count << " times".
  file.close();
  return 0;
}
```

Output :

Enter words to count : file
the word file occurs 0 times.

Q2
hello