St. Francis Institute of Technology
Borivli (West), Mumbai-400103
Department of Information Technology

Experiment – 1

1. Aim: Implementation of Object Oriented Concept of data encapsulation using C++

B. Program Code

Write and execute the following program codes in C++ to achieve the given aim and attach it with your own comments and with neat indentation.

1. WAP to create class item having the following details- two private members to store the fetched data from user. Two public functions get_data() and put_data() to fetch the data and display the fetched data. Access the class members from main by creating its objects

Program:-

```
#include<iostream>
using namespace std;

//Class item
class item{

private:
    //Private members of class item
    string name;
    int price;
public:
    //Public member functions of class item

    //Function to get values of item name and price
    void get_data(string name,int price){
        this -> name = name;
        this -> price = price;
    }
    //Function to display item name and item price
    void put_data(){
        cout<<"The name of item is: "<<name<<endl;
        cout<<"The price of item is: "<<price<<endl;
    }
};

int main(){

    //Initializing object of class item
    item i;
    //Function get_data called
```

```
    i.get_data("rice",40);
    //Function put_data called
    i.put_data();

    item j;
    j.get_data("bread",20);
    j.put_data();
    return 0;
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
Microsoft Windows [Version 10.0.18363.1139]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\College\PCPF\CPP\Exp1>Exp1-1
The name of item is: rice
The price of item is: 40
The name of item is: bread
The price of item is: 20

D:\College\PCPF\CPP\Exp1>
```

Analysis:
An object helps us to access the members of a class. A class can have multiple objects and we can decide the visibility/accessibility of the members of the class using the keywords public, private and protected.

2. WAP to perform shopping operations. The shopping operations should be able to perform Enter the price of item
Code number of the item
Provide an option to –add more items to the list, delete item from list, and print the total value of products.
Program:-

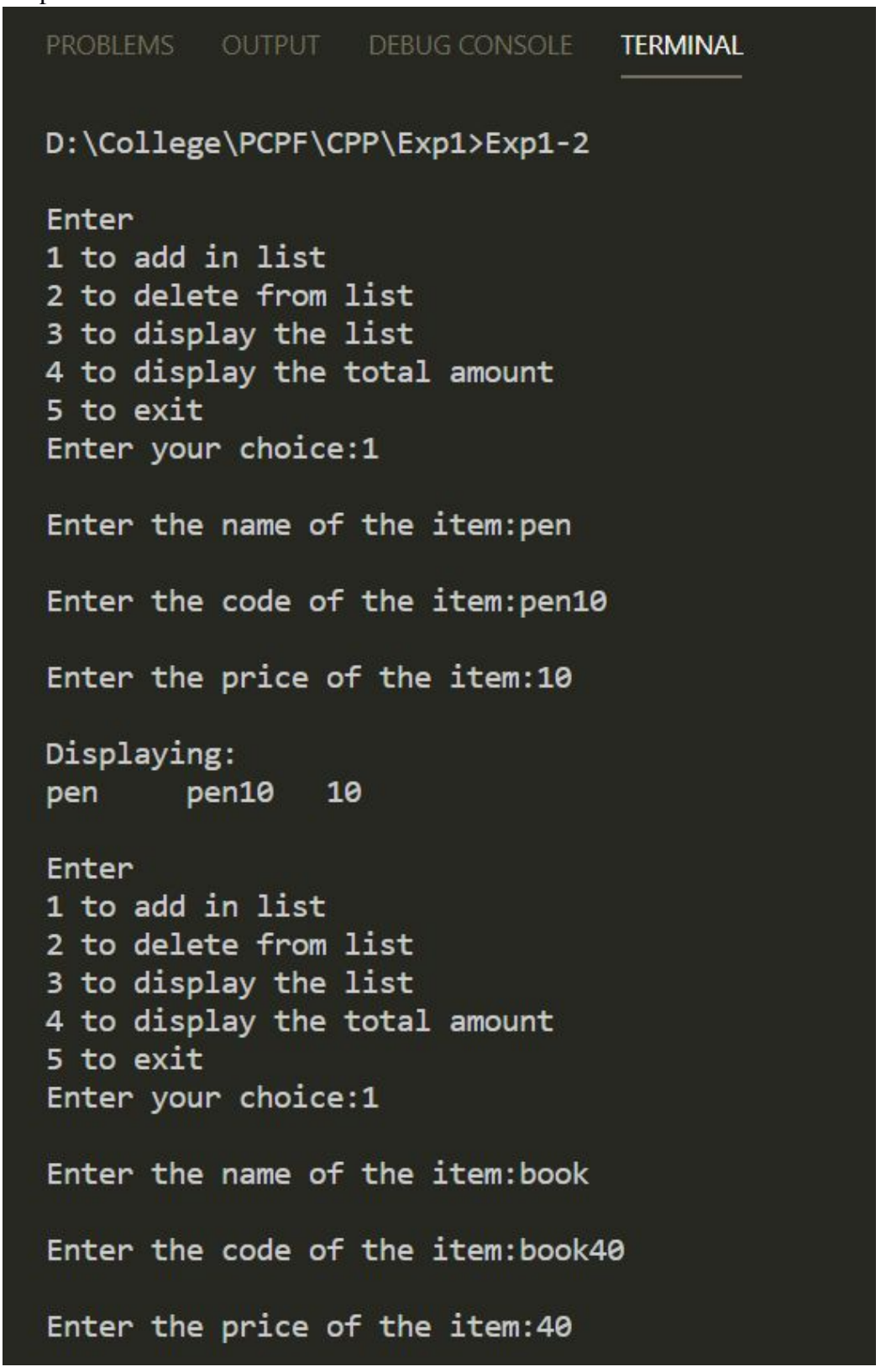
```
#include<iostream>
using namespace std;

class ShoppingList
{
private:
    //Declaring a singly linked list
    struct List{
        string name;
        string code;
        int price;
        struct List * next;
    };
    struct List * start = NULL;
```

```
public:
    //Declaring list functions
    void add_to_list(string iname, string icode, int iprice);
    void delete_from_list(string icode);
    void display();
    void display_total();
};

//To insert elements in the beginning of the list
void ShoppingList::add_to_list(string iname, string icode, int iprice)
{
    struct List * new_item = new List;
    new_item->name = iname;
    new_item->code = icode;
    new_item->price = iprice;
    new_item->next = start;
    start = new_item;
    display();
}

//To delete a given item of the list
void ShoppingList::delete_from_list(string icode)
{
    struct List * ptr, *preptr;
    ptr = start;
    if (ptr->code == icode)
    {
        start = start -> next;
    }
    else
    {
        while(ptr -> code != icode)
        {
            preptr = ptr;
            ptr = ptr -> next;
        }
        preptr -> next = ptr -> next;
    }
    cout<<"Deleted:"<<ptr->name<<ptr->code<<ptr->price<<endl;
    delete ptr;
    display();
}

//To display all the items in the list
```

```
void ShoppingList::display(){

    struct List * ptr;
    ptr = start;
    cout<<"Displaying:"<<endl;
    while (ptr != NULL)
    {
        cout<<ptr->name<<"\t"<<ptr->code<<"\t"<<ptr->price<<endl;
        ptr = ptr -> next;
    }

}

//To display the total amount
void ShoppingList::display_total(){

    struct List * ptr;
    ptr = start;
    int total=0;
    while (ptr != NULL)
    {
       total+=ptr->price;
        ptr = ptr -> next;
    }
    cout<<"Total price:"<<total<<endl;
}

int main(){

    //Initializing the object of class
    ShoppingList item;

    string name,code;
    int price,num;

    //Menu driven program
    do
    {
        cout<<"\nEnter"<<endl;
        cout<<"1 to add in list\n2 to delete from list\n3 to display the
list\n4 to display the total amount\n5 to exit"<<endl;
        cout<<"Enter your choice:";
        cin>>num;
        cout<<endl;
```

```
        switch (num)
        {
        case 1:
            cout<<"Enter the name of the item:";
            cin>>name;
            cout<<endl;
            cout<<"Enter the code of the item:";
            cin>>code;
            cout<<endl;
            cout<<"Enter the price of the item:";
            cin>>price;
            cout<<endl;
            item.add_to_list(name,code,price);
            break;
        case 2:
            cout<<"Enter the code of the item you want to delete:";
            cin>>code;
            item.delete_from_list(code);
            break;
        case 3:
            item.display();
            break;
        case 4:
            item.display_total();
            break;
        default:
            break;
        }
    } while (num<5);
    return 0;
}
```

Output:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
D:\College\PCPF\CPP\Exp1>Exp1-2
Enter
1 to add in list
2 to delete from list
3 to display the list
4 to display the total amount
5 to exit
Enter your choice:1
Enter the name of the item:pen
Enter the code of the item:pen10
Enter the price of the item:10
Displaying:
pen pen10 10
Enter
1 to add in list
2 to delete from list
3 to display the list
4 to display the total amount
5 to exit
Enter your choice:1
Enter the name of the item:book
Enter the code of the item:book40
Enter the price of the item:40

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Displaying:
book    book40  40
pen     pen10   10

Enter
1 to add in list
2 to delete from list
3 to display the list
4 to display the total amount
5 to exit
Enter your choice:1

Enter the name of the item:pouch

Enter the code of the item:pouch30

Enter the price of the item:30

Displaying:
pouch   pouch30 30
book    book40  40
pen     pen10   10

Enter
1 to add in list
2 to delete from list
3 to display the list
4 to display the total amount
5 to exit
Enter your choice:4

Total price:80
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

Enter
1 to add in list
2 to delete from list
3 to display the list
4 to display the total amount
5 to exit
Enter your choice:2

Enter the code of the item you want to delete:pen10
Deleted:penpen1010
Displaying:
pouch    pouch30 30
book     book40  40

Enter
1 to add in list
2 to delete from list
3 to display the list
4 to display the total amount
5 to exit
Enter your choice:5


D:\College\PCPF\CPP\Exp1>
```

Analysis:
In this program we declare a singly linked list to store the items in the cart. We use switch case to get the choices from the user. We have used the linked list operations of insertion, deletion and traversal to perform insertion of items, deletion of items, displaying the items and calculating the total.

3. Write a program to illustrate the use of object arrays. Create class employee having the following details- two private member data for storing name and age of employee. Create functions getdata( ) and putdata( ) to take the inputs and display the outputs

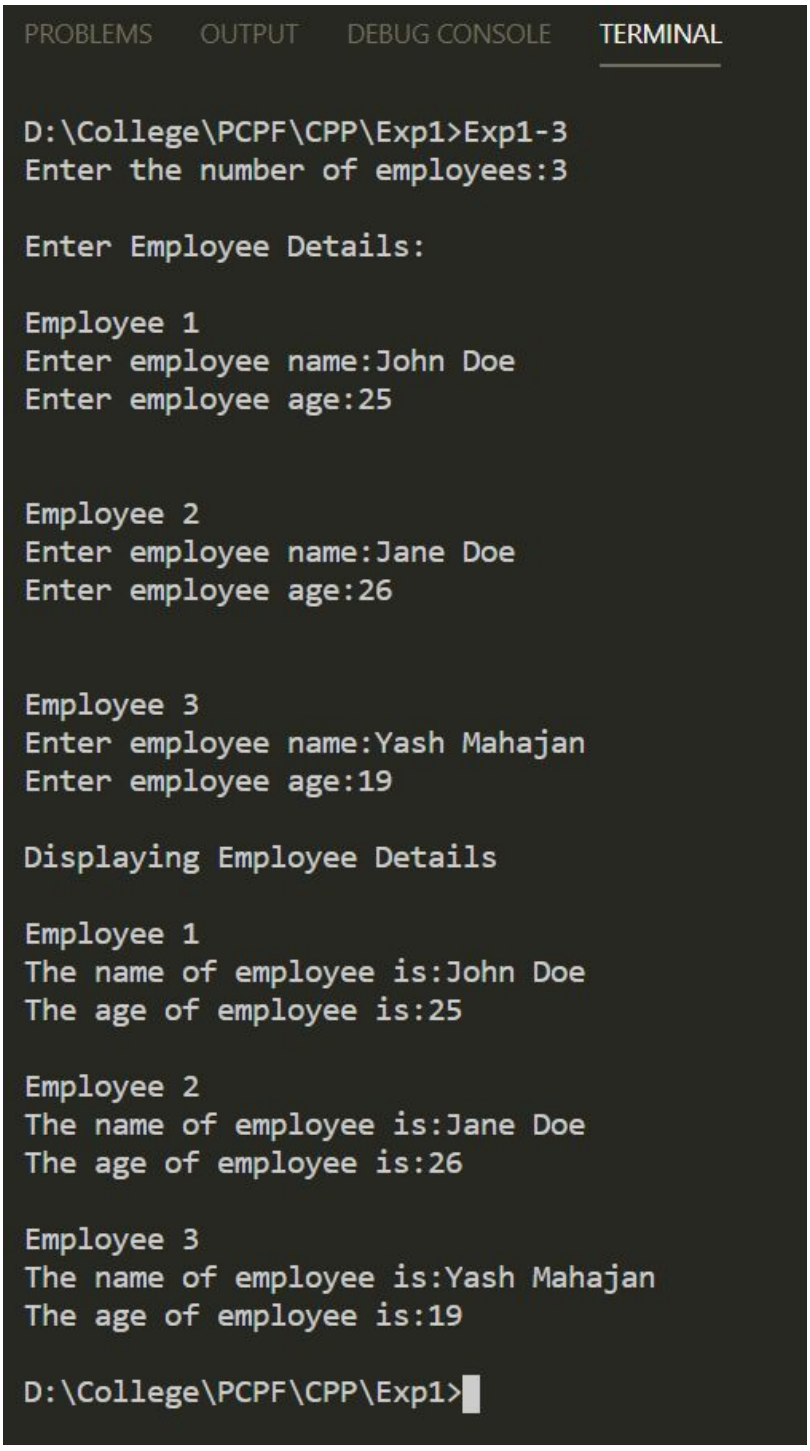
Program:-

```
#include<iostream>
using namespace std;
```

```
//Class Employee
class Employee
{
private:
    //Private members of the class Employee
    string name;
    int age;
public:
    //Public member functions of class Employee
    void getData(string name, int age);
    void putData();
};
//Function to get values of employee name and age
void Employee::getData(string name, int age){

    this-> name = name;
    this-> age = age;
}
//Function to display employee name and age
void Employee::putData(){

    cout<<"The name of employee is:"<<name<<endl;
    cout<<"The age of employee is:"<<age<<endl;

}

int main(){

    string name;
    int n, age;
    cout<<"Enter the number of employees:";
    cin>>n;
    cout<<endl;
    cout<<"Enter Employee Details:"<<endl;

    //Initializing  object array of class Employee
    Employee emp[n];

    for (int i = 0; i < sizeof(emp)/sizeof(emp[0]); i++)
    {
        cout<<"\nEmployee "<<i+1<<endl;
        cout<<"Enter employee name:";
        cin.ignore();
        getline(cin, name);
```

```
        cout<<"Enter employee age:";
        cin>>age;
        cout<<endl;
        //Function getData() called
        emp[i].getData(name,age);
    }
    cout<<"Displaying Employee Details"<<endl;
    for (int i = 0; i < sizeof(emp)/sizeof(emp[0]); i++)
    {
        cout<<"\nEmployee "<<i+1<<endl;
        //Function putData() called
        emp[i].putData();
    }
    return 0;

}
```

Output:



Analysis:

The array of type class contains the objects of the class as its individual elements. Thus, an array of a class type is also known as an array of objects. An array of objects is declared in the same way as an array of any built-in data type.

8. Post Experimental Exercise

A. Questions:

1. What is a class? How does it accomplish data hiding?

A class is a model for a set of objects. It establishes what its data will be(type together with their visibility) and fixes their name, signature, visibility and implementation for each of its

methods.
Data hiding in a class is achieved by specifying the visibility access of the data member. There are 3 visibility methods public, protected and private.
A public member is accessible from anywhere outside the class but within a program.
A private member variable or function cannot be accessed, or even viewed from outside the class. Only the class and friend functions can access private members.
A protected member variable or function is very similar to a private member but it provides one additional benefit that they can be accessed in child classes which are called derived classes.

```
class Base {
public:
  // public members go here
protected:

  // protected members go here
private:
  // private members go here
};
```

2. What are objects? How are they created?

An object can be a variable, a data structure, a function, or a method, and as such, is a value in memory referenced by an identifier.
An object is an instance of a class and we can use objects to access the data members and member functions of a class.
When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.
The syntax to declare an object in C++ is
class_name object_name;
Animal dog;
Here dog is an object of class Animal.

3. What is a static class? When do we declare a member of a class as static?

A static class is similar to a non static class, the only difference being that an object of a static class cannot be instantiated. C++ does not support static class but it supports static data members and member functions.
Classes can contain static member data and member functions. When a data member is declared as **static**, only one copy of the data is maintained for all objects of the class. Static data members are not part of objects of a given class type. As a result, the declaration of a static data member is not considered a definition. The data member is declared in class scope, but definition is performed at file scope. These static members have external linkage. We declare a member of a class static when we want to have a single copy of that member throughout the program.

4. What is a friend function? What are the merits and demerits of using a friend function?

A friend function of a class is defined outside that class' scope but it has the right to access all private and protected members of the class. Even though the prototypes for friend functions appear in the class definition, friends are not member functions.
A friend can be a function, function template, or member function, or a class or class template, in which case the entire class and all of its members are friends.
To declare a function as a friend of a class, precede the function prototype in the class definition with keyword **friend.**
Merits of friend function are:-
1. It acts as the bridge between two classes by operating on their private data.
2. It is able to access members without need of inheriting the class.

3. It can be used to increase the versatility of overloading operators.
4. It provides functions that need data which isn't normally used by the class.
5. Allows sharing private class information by a non-member function.

Demerits of friend function are:-
1. It violates the law of data hiding by allowing access to private members of the class from outside the class.
2. Breach of data integrity.
3. Conceptually messy
4. Runtime polymorphism in the member cannot be done.
5. Size of memory occupied by objects will be maximum.

5. WAP to define a class to represent bank account. Include the following members
Data members
Name of depositor, account number, type of account, balance amt. in account Member functions
To assign initial values, to deposit an amount, to withdraw an amount after checking the balance, to display name and balance

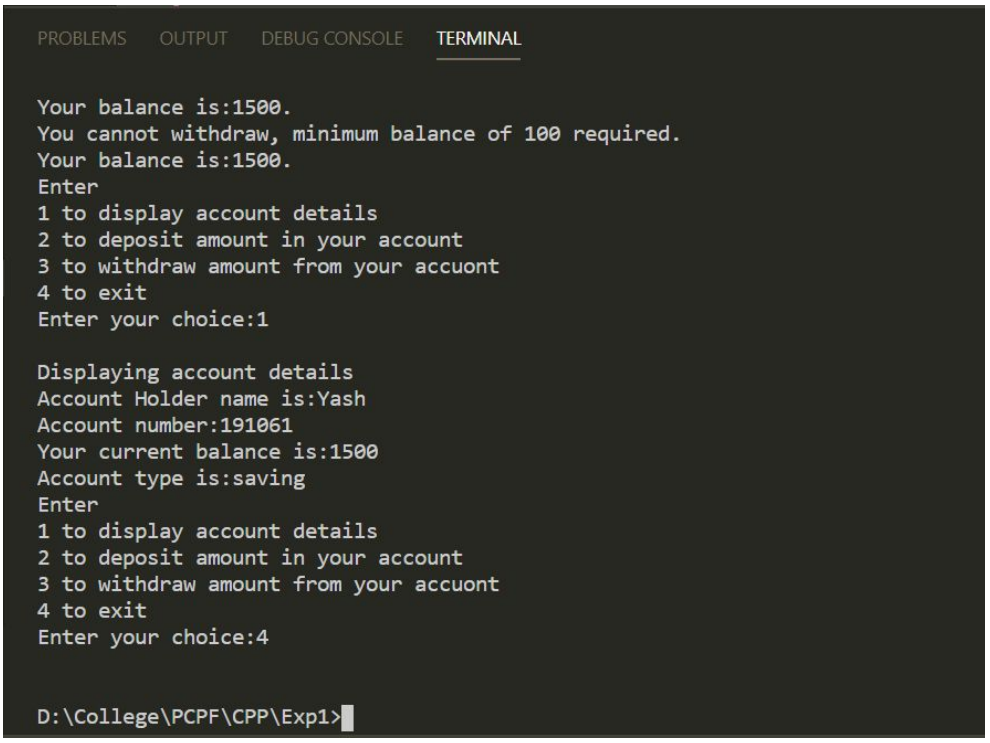
```
#include<iostream>
using namespace std;

//Class declaration
class Account{

//Private data members
private:
    string name;
    float balance;
    int account_number;
    string type;

//public member functions
public:
    void setData(string name, float balance, int account_number,
string type);
    void deposit(float amount);
    void withdraw(float amount);
    void display();
};

//Function to set values of depositor name, balance, account number
and account type given by user
```

```
void Account::setData(string name, float balance, int
account_number, string type){

    this-> name = name;
    this-> balance = balance;
    this-> account_number = account_number;
    this-> type = type;
}

//Function to deposit amount to the account
void Account::deposit(float amount){

    if (amount>0)
    {
        balance+=amount;
        cout<<amount<<" deposited."<<endl;
    }
    else
    {
        cout<<"Enter valid amount."<<endl;
    }
    cout<<"Your balance is:"<<balance<<"."<<endl;
}

//Function to withdraw amount from the account
void Account::withdraw(float amount){

    cout<<"Your balance is:"<<balance<<"."<<endl;
    if (amount>0 && (balance-amount)>=100) //if amount to be
withdrawn is greater than 0 and the difference between balance and
amount is greater than minimum balance then withdraw
    {
        balance-=amount;
        cout<<amount<<" withdrawn."<<endl;
    }
    else if (balance-amount<100)//minimum balance of 100 required
message
    {
        cout<<"You cannot withdraw, minimum balance of 100
required."<<endl;
    }
    else //incorrect amount entered
    {
        cout<<"Enter valid amount."<<endl;
```

```
    }
    cout<<"Your balance is:"<<balance<<"."<<endl;
}

void Account::display(){

    cout<<"Displaying account details"<<endl;
    cout<<"Account Holder name is:"<<name<<endl;
    cout<<"Account number:"<<account_number<<endl;
    cout<<"Your current balance is:"<<balance<<endl;
    cout<<"Account type is:"<<type<<endl;
}

int main(){

    int num, account_number;
    string name, type;
    float balance, amount;

    Account a;
    cout<<"Enter Bank details:"<<endl;
    cout<<"Enter Your name, account number, balance, account
type:";
    cin>>name>>account_number>>balance>>type;
    cout<<endl;
    a.setData(name,balance,account_number,type);
    do
    {   cout<<"Enter"<<endl<<"1 to display account
details"<<endl<<"2 to deposit amount in your account"<<endl<<"3 to
withdraw amount from your account"<<endl<<"4 to exit"<<endl;
        cout<<"Enter your choice:";
        cin>>num;
        cout<<endl;
        switch (num)
        {
        case 1:
            a.display();
            break;
        case 2:
            cout<<"Enter the amount to be deposited:";
            cin>>amount;
            cout<<endl;
            a.deposit(amount);
            break;
```

```
        case 3:
             cout<<"Enter the amount to be withdrawn:";
             cin>>amount;
             cout<<endl;
             a.withdraw(amount);
             break;

        default:
             break;
        }
    } while (num<4);


    return 0;

}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

D:\College\PCPF\CPP\Exp1>Question_5
Enter Bank details:
Enter Your name, account number, balance, account type:Yash 191061 2000 saving

Enter
1 to display account details
2 to deposit amount in your account
3 to withdraw amount from your accuont
4 to exit
Enter your choice:2

Enter the amount to be deposited:1000

1000 deposited.
Your balance is:3000.
Enter
1 to display account details
2 to deposit amount in your account
3 to withdraw amount from your accuont
4 to exit
Enter your choice:3

Enter the amount to be withdrawed:1500

Your balance is:3000.
1500 withdrawed.
Your balance is:1500.
Enter
1 to display account details
2 to deposit amount in your account
3 to withdraw amount from your accuont
4 to exit
Enter your choice:3

Enter the amount to be withdrawed:1450
```



C. Conclusion:

Object Oriented programming (OOP) is a programming paradigm that relies on the concept of classes and objects. In this experiment we have written programs in C++ to implement concepts of object oriented programming like classes objects, data members and member functions and the concept of Data Encapsulation and visibility access modes.

To perform this experiment Visual Studio Code was used as a code editing software and MinGW GCC compiler was used to compile the codes

From this experiment we can infer that a class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects. An object is an instance of a class and we can use objects to access the data members and member functions of a class. Data encapsulation refers to the process of bundling data with the methods that operate on the data. Classes help us to make the code reusable and helps us avoid repetition of code. Encapsulation allows us to make changes independently without having to change the entire code, making the code easily maintainable.

D. References:

[1] E Balaguruswamy, “Object Oriented Programming with C++”, second edition Tata McGraw Hill (Chapter 5)