DAY 4

Accounts

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

class Account
{
  int accountNumber;
  int balance;

public:
  void setAccount(int accountNumber, int balance)
  {
    this->accountNumber = accountNumber;
    this->balance = balance;
  }

  void displayAccounts() const
  {
    cout << "Account number: " << accountNumber << endl;
    cout << "Balance: " << balance << endl;
  }

  void transferByValue(Account account1, Account account2, int amount)
  {
    if (account1.balance < amount)
    {
      cout << "Insufficient balance" << endl;
      return;
    }
    int account1Balance = account1.balance - amount;
    int account2Balance = account2.balance + amount;
    account1.balance = account1Balance;
    account2.balance = account2Balance;

    cout << "Account 1 balance: " << account1Balance << endl;
    cout << "Account 2 balance: " << account2Balance << endl;
    cout << "Amount transferred: " << amount << endl;
    cout << "Transfer by value successful with changes in local
variables" << endl;
  }

  void transferByReference(Account &account1, Account &account2, int
amount)
  {
    if (account1.balance < amount)
    {
```

```
      cout << "Insufficient balance" << endl;
      return;
    }
    account1.balance -= amount;
    account2.balance += amount;

    cout << "Account 1 balance: " << account1.balance << endl;
    cout << "Account 2 balance: " << account2.balance << endl;
    cout << "Amount transferred: " << amount << endl;
    cout << "Transfer by reference successful" << endl;
  }

  void transferByAddress(Account *account1, Account *account2, int
amount)
  {
    if (account1->balance < amount)
    {
      cout << "Insufficient balance" << endl;
      return;
    }
    account1->balance -= amount;
    account2->balance += amount;

    cout << "Account 1 balance: " << account1->balance << endl;
    cout << "Account 2 balance: " << account2->balance << endl;
    cout << "Amount transferred: " << amount << endl;
    cout << "Transfer by address successful" << endl;
  }

  int getAccountNumber() const
  {
    return accountNumber;
  }
};

int main()
{
  int n;
  cout << "Enter the number of accounts: ";
  cin >> n;

  Account *accounts = new Account[n];

  for (int i = 0; i < n; i++)
  {
    int accountNumber, balance;
    cout << "Enter account number: ";
    cin >> accountNumber;
```

```
    cout << "Enter balance: ";
    cin >> balance;
    accounts[i].setAccount(accountNumber, balance);
  }

  while (true)
  {
    cout << "1. Transfer by value" << endl;
    cout << "2. Transfer by reference" << endl;
    cout << "3. Transfer by address" << endl;
    cout << "4. Display the Accounts" << endl;
    cout << "5. Exit" << endl;
    cout << "Enter your choice: ";
    int choice;
    cin >> choice;

    switch (choice)
    {
    case 1:
    {
      int account1Number, account2Number, amount;
      cout << "Enter account number 1: ";
      cin >> account1Number;
      cout << "Enter account number 2: ";
      cin >> account2Number;
      cout << "Enter amount: ";
      cin >> amount;

      Account *account1;
      Account *account2;
      for (int i = 0; i < n; i++)
      {
        if (accounts[i].getAccountNumber() == account1Number)
        {
          account1 = &accounts[i];
        }
        if (accounts[i].getAccountNumber() == account2Number)
        {
          account2 = &accounts[i];
        }
      }
      if (account1 && account2)
      {
        accounts[0].transferByValue(*account1, *account2, amount);
      }
      else
      {
        cout << "One or both account numbers are invalid." << endl;
```

```
    }
    break;

  }
  case 2:
  {
    int account1Number, account2Number, amount;
    cout << "Enter account number 1: ";
    cin >> account1Number;
    cout << "Enter account number 2: ";
    cin >> account2Number;
    cout << "Enter amount: ";
    cin >> amount;

    Account *account1;
    Account *account2;
    for (int i = 0; i < n; i++)
    {
      if (accounts[i].getAccountNumber() == account1Number)
      {
        account1 = &accounts[i];
      }
      if (accounts[i].getAccountNumber() == account2Number)
      {
        account2 = &accounts[i];
      }
    }
    if (account1 && account2)
    {
      accounts[0].transferByReference(*account1, *account2, amount);
    }
    else
    {
      cout << "One or both account numbers are invalid." << endl;
    }
    break;
  }
  case 3:
  {
    int account1Number, account2Number, amount;
    cout << "Enter account number 1: ";
    cin >> account1Number;
    cout << "Enter account number 2: ";
    cin >> account2Number;
    cout << "Enter amount: ";
    cin >> amount;

    Account *account1;
```

```
      Account *account2;
      for (int i = 0; i < n; i++)
      {
        if (accounts[i].getAccountNumber() == account1Number)
        {
          account1 = &accounts[i];
        }
        if (accounts[i].getAccountNumber() == account2Number)
        {
          account2 = &accounts[i];
        }
      }
      if (account1 && account2)
      {
        accounts[0].transferByAddress(account1, account2, amount);
      }
      else
      {
        cout << "One or both account numbers are invalid." << endl;
      }
      break;
    }
    case 4:
    {
      for (int i = 0; i < n; i++)
      {
        accounts[i].displayAccounts();
      }
      break;
    }
    case 5:
    {
      cout << "Exiting..." << endl;
      delete[] accounts;
      return 0;
    }
    default:
    {
      cout << "Invalid choice" << endl;
    }
    }
  }

  delete[] accounts;

  return 0;
}
```

Constructor Dead Alive

```
#include <iostream>
#include <string.h>

using namespace std;

class MyClass
{
private:
  string name;
  static int alive, dead, total;

public:
  MyClass(const string &objName) : name(objName)
  {
    alive++;
    total++;
    cout << "Constructor called for object: " << name << endl;
    cout << "Alive: " << alive << endl;
    cout << "Dead: " << dead << endl;
    cout << "Total: " << total << endl;
  }

  ~MyClass()
  {
    alive--;
    dead++;
    cout << "Destructor called for object: " << name << endl;
    cout << "Alive: " << alive << endl;
    cout << "Dead: " << dead << endl;
    cout << "Total: " << total << endl;
  }
};
int MyClass::alive = 0;
int MyClass::dead = 0;
int MyClass::total = 0;
int main()
{
  cout << "Creating object A\n";
  MyClass objA("A");

  {
    cout << "Creating object B inside a block\n";
    MyClass objB("B");
    cout << "Exiting the block." << endl;
  }

  cout << "Creating object C\n";
```

```
  MyClass objC("C");
  cout << "Exiting the main." << endl;
  return 0;
}
```

Template Multiplier

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

template <typename T, typename U>
class Multiplier
{
  T value1;
  U value2;
public:
  void setValues()
  {
    cout << "Enter value 1: ";
    cin >> value1;
    cout << "Enter value 2: ";
    cin >> value2;
  }

  void multiply()
  {
    cout << "Multiplication of " << value1 << " and " << value2 << " is:
" << value1 * value2 << endl;
  }
};

int main()
{
  Multiplier<int, double> MultiplyIntDouble;
  MultiplyIntDouble.setValues();
  MultiplyIntDouble.multiply();
  return 0;
}
```

Publication

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

class Publication
{
```

```
private:
  string title;
  int price;

public:
  Publication() : title("UNTITLED"), price(0) {}

  void setTitle(string name)
  {
    title = name;
  }

  void setPrice(int n)
  {
    price = n;
  }

  string getTitle()
  {

    return title;
  }
  int getPrice()
  {
    return price;
  }
};

class Sales
{
private:
  int soldCopy;
  int printedCopy;

public:
  Sales() : soldCopy(0), printedCopy(0) {}
  void setSoldCopy(int n)
  {
    soldCopy = n;
  }

  void setPrintedCopy(int n)
  {
    printedCopy = n;
  }
  int getSoldCopy()
  {
    return soldCopy;
```

```
  }
  int getPrintedCopy()
  {
    return printedCopy;
  }
};

class Book
{
  Publication pub;
  Sales sal;
  int totalPages;

public:
  Book() : pub(), sal(), totalPages(0) {}

  Book(string title, int price, int printedCopy, int soldCopy, int pages)
: totalPages(pages)
  {
    pub.setTitle(title);
    pub.setPrice(price);
    sal.setPrintedCopy(printedCopy);
    sal.setSoldCopy(soldCopy);
  }

  void display()
  {
    cout << "Title: " << pub.getTitle() << endl;
    cout << "Price: " << pub.getPrice() << endl;
    cout << "Number of copies printed: " << sal.getPrintedCopy() << endl;
    cout << "Number of copies sold: " << sal.getSoldCopy() << endl;
    cout << "Number of pages: " << totalPages << endl;
  }
};

int main()
{
  Book b1;
  b1.display();
  cout << endl;
  string title;
  int price, soldCopy, pages, printedCopy;

  cout << "Enter the title of the book:";
  cin >> title;
  cout << "Enter the price of the book:";
  cin >> price;
  cout << "Enter the number of copies printed:";
```

```
  cin >> printedCopy;
  cout << "Enter the number of copies sold:";
  cin >> soldCopy;
  cout << "Enter the number of pages in the book:";
  cin >> pages;

  Book b2(title, price, printedCopy, soldCopy, pages);
  b2.display();
}
```

Inheritance Publication

```
#include <iostream>
#include <string>

using namespace std;

class Publication
{
private:
  string title;
  int price;

public:
  Publication() : title("UNTITLED"), price(0) {}

  void setTitle(string name)
  {
    title = name;
  }

  void setPrice(int n)
  {
    price = n;
  }

  string getTitle()
  {

    return title;
  }
  int getPrice()
  {
    return price;
  }
};

class Sales
```

```
{
private:
  int soldCopy;
  int printedCopy;

public:
  Sales() : soldCopy(0), printedCopy(0) {}
  void setSoldCopy(int n)
  {
    soldCopy = n;
  }
  void setPrintedCopy(int n)
  {
    printedCopy = n;
  }
  int getSoldCopy()
  {
    return soldCopy;
  }
  int getPrintedCopy()
  {
    return printedCopy;
  }
};

class Book : public Publication, public Sales
{
private:
  int pages;

public:
  Book() : Publication(), Sales(), pages(0) {}
  Book(string title, int price, int soldCopy, int printedCopy, int pages)
: Publication(), Sales(), pages(pages)
  {
    setTitle(title);
    setPrice(price);
    setSoldCopy(soldCopy);
    setPrintedCopy(printedCopy);
  }

  void displayBook()
  {
    cout << "Title: " << getTitle() << endl;
    cout << "Price: " << getPrice() << endl;
    cout << "Sold copies: " << getSoldCopy() << endl;
    cout << "Printed copies: " << getPrintedCopy() << endl;
    cout << "Pages: " << pages << endl;
```

```
  }
};

int main()
{
  Book book1;
  book1.displayBook();

  cout << endl;

  string title;
  int price, soldCopy, printedCopy, pages;

  cout << "Enter the title of the book: ";
  cin >> title;
  cout << "Enter the price of the book: ";
  cin >> price;
  cout << "Enter the number of sold copies: ";
  cin >> soldCopy;
  cout << "Enter the number of printed copies: ";
  cin >> printedCopy;
  cout << "Enter the number of pages: ";
  cin >> pages;

  Book book2(title, price, soldCopy, printedCopy, pages);
  book2.displayBook();
  return 0;
}
```

Question Answer

```
#include <iostream>
#include <stdarg.h>
using namespace std;

int totalScore(int n, ...)
{
  va_list args;
  va_start(args, n);
  int firstHighest = 0;
  int secondHighest = 0;
  int thirdHighest = 0;

  for (int i = 0; i < n; i++)
  {
    int marks = va_arg(args, int);
    if (marks > firstHighest)
    {
```

```
      thirdHighest = secondHighest;
      secondHighest = firstHighest;
      firstHighest = marks;
    }
    else if (marks > secondHighest)
    {
      thirdHighest = secondHighest;
      secondHighest = marks;
    }
    else if (marks > thirdHighest)
    {
      thirdHighest = marks;
    }
  }
  int total = firstHighest + secondHighest + thirdHighest;

  va_end(args);

  return total;
}

int main()
{
  int questions[5];
  int marks;
  int n;
  cout << "Enter the number of questions attempted: ";
  cin >> n;
  if (n < 3)
  {
    cout << "Number of questions attempted cannot be less than 3" <<
endl;
    return 0;
  }
  if (n > 5)
  {
    cout << "Number of questions attempted cannot be more than 5" <<
endl;
    return 0;
  }
  for (int i = 0; i < n; i++)
  {
    cout << "Enter the marks of question(0-15) " << i + 1 << ": ";
    cin >> questions[i];
    if (questions[i] < 0 || questions[i] > 15)
    {
      cout << "Marks should be between 0 and 15" << endl;
      return 0;
```

```
    }
  }
  int result = totalScore(n, questions[0], questions[1], questions[2],
questions[3], questions[4]);

  cout << "Total score: " << result << endl;
}
```