

**JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY**

**COLLEGE OF ENGINEERING AND TECHNOLOGY**

**SCHOOL OF ELECTRICAL, ELECTRONICS AND INFORMATION ENGINEERING**

**DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING**

**B.SC ELECTRICAL AND ELECTRONICS ENGINEERING**

**UNIT TITLE: COMPUTER PROGRAMMING II**

**UNIT CODE: ICS 2276**

**CAT 2**

**GROUP MEMBERS**

|  |  |  |
| --- | --- | --- |
| **NAME** | **REGISTRATION NUMBER** | **SIGN** |
| DENNIS MUENDO | ENE211-0025/2020 |  |
| JAMES WAIGWA | ENE211-0047/2020 |  |
| ELLERTONE ONGOKI | ENE211-0055/2020 |  |
| REAGAN SIMIYU | ENE211-0279/2020 |  |

**QUESTION 1**

Booker University library maintains an inventory of books. The list includes the details: Author, price, title, book\_number and number of copies of each book. Whenever new books are purchased the librarian adds the book details into the database. The chief librarian occasionally requests for a list of all the books in the database.

Required

.Construct a simple class called book with suitable data members and member functions to:-

i) .Insert a new book record into the database

ii). Display a list of all books in the database

iii). Write a main function XE "function" to test the program

CODE

#include <iostream>

#include <string>

using namespace std;

class Book {

private:

string author;

double price;

string title;

int book\_number;

int num\_copies;

public:

void insert(string a, double p, string t, int bn, int nc) {

author = a;

price = p;

title = t;

book\_number = bn;

num\_copies = nc;

}

void display() {

cout << "Author: " << author << endl;

cout << "Price: " << price << endl;

cout << "Title: " << title << endl;

cout << "Book Number: " << book\_number << endl;

cout << "Number of Copies: " << num\_copies << endl;

}

};

int main() {

Book books[10]; // Increased size to hold more books

int num\_books = 0; // Keep track of the number of books currently in the database

int choice;

while (true) {

cout << "Select an option: " << endl;

cout << "1. Insert a book" << endl;

cout << "2. View all books" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

if (choice == 1) {

if (num\_books >= 10) {

cout << "Database is full. Cannot insert more books." << endl;

} else {

string author, title;

double price;

int book\_number, num\_copies;

cin.ignore(); // Ignore any leftover newline characters

cout << "Enter author for book " << num\_books+1 << ": ";

getline(cin, author);

cout << "Enter title for book " << num\_books+1 << ": ";

getline(cin, title);

cout << "Enter price for book " << num\_books+1 << ": ";

cin >> price;

cout << "Enter book number for book " << num\_books+1 << ": ";

cin >> book\_number;

cout << "Enter number of copies for book " << num\_books+1 << ": ";

cin >> num\_copies;

cin.ignore(); // Ignore any leftover newline characters

books[num\_books].insert(author, price, title, book\_number, num\_copies);

num\_books++;

cout<<"Book added to the database." << endl;

}

} else if (choice == 2) {

if (num\_books == 0) {

cout << "Database is empty. No books to display." << endl;

} else {

cout << "List of all books in the database: " << endl;

for (int i = 0; i < num\_books; i++) {

books[i].display();

}

}

} else if (choice == 3) {

break; // Exit the loop and end the program

} else {

cout << "Invalid choice. Please enter a number between 1 and 3." << endl;

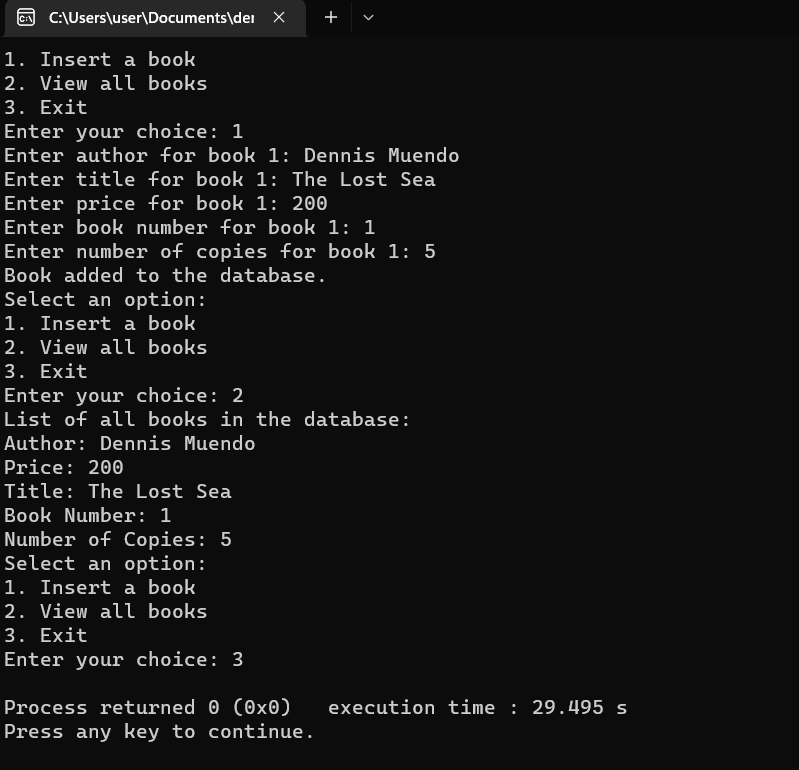
}

}

return 0;

}

**OUTPUT**

****

**QUESTION 2**

**2. DT Dobie (K) Ltd sells a range of vehicles, including Mercedes and Nissan in the East African region. Vehicle sale details include the Make (e.g. Nissan), Model (e.g. Sunny), Engine Number and Sale Price.**

1. **Create a class vehicle that captures the above data using a function called set\_vehicle()**
2. **The company makes a profit of 15% from the sale price for every vehicle. Create another function called get\_profit() which is still a member of the class vehicle to calculate and return the profit**
3. **Implement (i) and (ii) using an object in the main() function to capture the vehicle details and display the profit. In the output you should able to change the make model and those other details in order to get different profits for different amounts of money in the input**

**CODE**

#include <iostream>

using namespace std;

class Vehicle {

private:

string make;

string model;

string engine\_number;

int sale\_price;

public:

void set\_vehicle(string make, string model, string engine\_number, int sale\_price) {

this->make = make;

this->model = model;

this->engine\_number = engine\_number;

this->sale\_price = sale\_price;

}

int get\_profit() {

int profit = 0.15 \* sale\_price;

return profit;

}

};

int main() {

// create an object of Vehicle class

Vehicle v;

// get vehicle details from user input

string make, model, engine\_number;

int sale\_price;

cout << "Enter the make of the vehicle: ";

getline(cin, make);

cout << "Enter the model of the vehicle: ";

getline(cin, model);

cout << "Enter the engine number of the vehicle: ";

getline(cin, engine\_number);

cout << "Enter the sale price of the vehicle: ";

cin >> sale\_price;

// set vehicle details using user input

v.set\_vehicle(make, model, engine\_number, sale\_price);

// get profit

int profit = v.get\_profit();

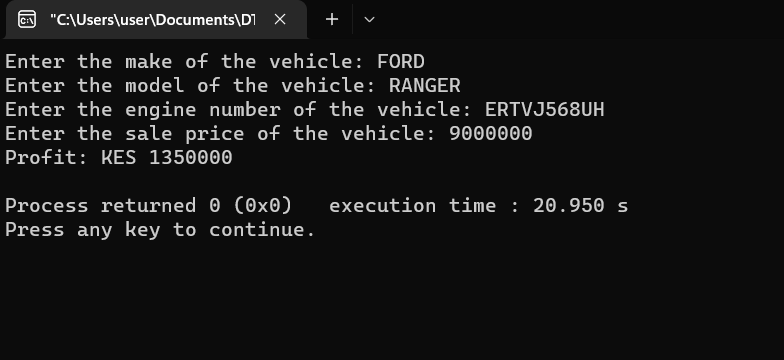
// display profit

cout << "Profit: KES " << profit << endl;

return 0;

}

**OUTPUT**

****

**QUESTION 3**

The Interim Independent Electoral Commission (IIEC) of Kenya requires an Electronic Voting Management System (EVMS) through which new voters can be added, invalid voters deleted as well as displaying of a voter’s details, among other functions.

Voter details include voter card ID, National ID Number, First Name, middle name, surname, polling station, date of birth (entered in dd-mm-yyyy format) and gender .

In an interview for an advertised Software Developer position, the commission has requested you to demonstrate with a simple program how a voter details can be added and displayed. Using your own defined and appropriate voter class and an interactive driver program (i.e. main function) write a C++ program to achieve this.

**CODE**

# define a class for Voter with attributes and methods

import uuid

import re

class Voter:

def \_\_init\_\_(self, voter\_card\_id, national\_id, first\_name, middle\_name, last\_name, polling\_station, date\_of\_birth, gender):

self.voter\_card\_id = voter\_card\_id

self.national\_id = national\_id

self.first\_name = first\_name

self.middle\_name = middle\_name

self.last\_name = last\_name

self.polling\_station = polling\_station

self.date\_of\_birth = date\_of\_birth

self.gender = gender

def display\_voter\_details(self):

# display the details of a voter

print("Voter Card ID: ", self.voter\_card\_id)

print("National ID: ", self.national\_id)

print("First Name: ", self.first\_name)

print("Middle Name: ", self.middle\_name)

print("Last Name: ", self.last\_name)

print("Polling Station: ", self.polling\_station)

print("Date of Birth: ", self.date\_of\_birth)

print("Gender: ", self.gender)

# create an empty list to store the voters

voter\_list = []

test = Voter("00001", "1000", "Uhuru", "Mwigai", "Kenyatta", "Gatundu", "12-12-1963", "Male")

voter\_list.append(test)

# Interactive driver program

while True:

# display menu

print("\nWelcome to The Interim Independent Electoral Commission (IIEC)")

print("Electronic Voting Management System (EVMS)")

print("\nSelect an option:")

print("1. Add new voter details")

print("2. Display voter details")

print("3. Delete voter details")

print("4. Exit")

choice = input("Enter your choice: ")

# add new voter details

if choice == '1':

print("\nEnter the following voter details:")

voter\_card\_id = str(uuid.uuid4())[:5]

national\_id = input("National ID: ")

first\_name = input("First Name: ")

middle\_name = input("Middle Name: ")

last\_name = input("Last Name: ")

polling\_station = input("Polling Station: ")

date\_of\_birth = input("Date of Birth (dd-mm-yyyy): ")

# check for valid date format

while not re.match(r'\d{2}-\d{2}-\d{4}', date\_of\_birth):

date\_of\_birth = input("Invalid format. Please enter your Date of Birth in the format dd-mm-yyyy: ")

gender = input("Gender: ")

# create a new Voter object and append to the list

voter = Voter(voter\_card\_id, national\_id, first\_name, middle\_name, last\_name, polling\_station, date\_of\_birth, gender)

voter\_list.append(voter)

print("\nVoter details added successfully.")

# display voter details

elif choice == '2':

if voter\_list: # check if list is not empty

# display voter details of all voters in list

for voter in voter\_list:

print("\nVoter details")

print("===================")

voter.display\_voter\_details()

else:

print("\nNo voter details found. Please add voter details first.")

#delete voter details

elif choice == '3':

if voter\_list: # check if list is not empty

voter\_card\_id = input("\nEnter the Voter Card ID of the voter whose details you want to delete: ")

for voter in voter\_list:

if voter.voter\_card\_id == voter\_card\_id:

voter\_list.remove(voter)

print("\nVoter details deleted successfully.")

break

else:

print("\nVoter with Voter Card ID '{}' not found.".format(voter\_card\_id))

else:

print("\nNo voter details found. Please add voter details first.")

# exit the program

elif choice == '4':

print("\nExiting the program...")

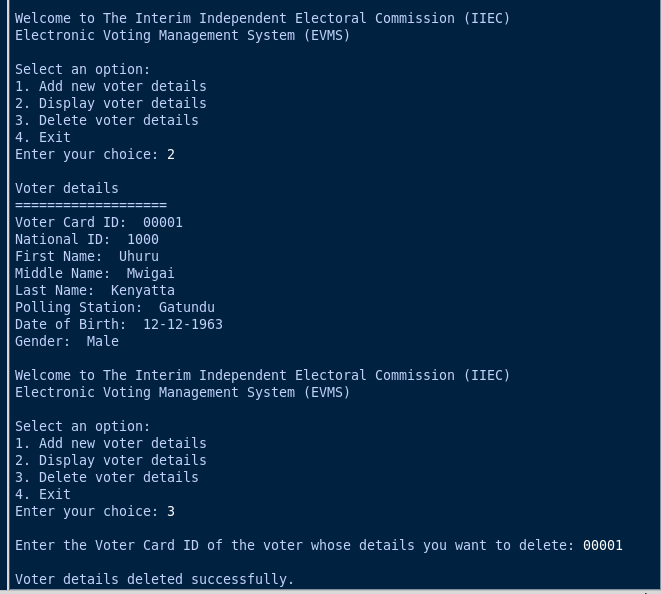
break

# invalid choice

else:

print("\nInvalid choice. Please select a valid option.")

**OUTPUTS**



**Question 4 {10 Marks}**

Safaricom Ltd intends to develop an Electronic Reward system aimed at rewarding its subscribers with Bonus (“Bonga”) points, where subscribers can redeem points for airtime or prizes. The system shall track the amount of air time a subscriber has and reward as follows:

**Airtime Bonus Points**

Equal or above Kshs. 2000.00 500

Ksh. 1,000.00 - Ksh 1,999.00 300

Ksh. 500.000 - Ksh 999.00 100

Ksh. 100.000 - Ksh 499.00 50

Below Ksh. 100.00 0

Write a C++ program that captures the Subscriber name, Phone number and Air time amount, through a constructor, uses a function compute\_bonuspoints() to calculate the points awarded, then outputs information as follows:

**Code**

#include <iostream>

#include <string>

using namespace std;

class Subscriber {

public:

// member initialization list to set the values of class members

Subscriber(string name, string phone, double airtime, double bonus\_points): name(name), phone(phone), airtime(airtime), bonus\_points(bonus\_points)

{}

// Declare the parameters and member variables as double

void compute\_bonuspoints() {

if (airtime >= 2000.0) {

bonus\_points = 500.0;

} else if (airtime >= 1000.0) {

bonus\_points = 300.0;

} else if (airtime >= 500.0) {

bonus\_points = 100.0;

} else if (airtime >= 100.0) {

bonus\_points = 50.0;

} else {

bonus\_points=0.0;

}

}

void print\_info(){

cout << name << ": (PHONE NO:" << phone << "): AWARDED " << bonus\_points << " BONGA POINTS. STAY WITH SAFARICOM. THE BETTER OPTION!" << endl;

}

private:

string name;

string phone;

double airtime;

double bonus\_points;

};

int main()

{

string name, phone; // Declare name and phone variables

double airtime,bonus\_points;

cout << "Enter customer's name: ";

getline(cin, name);

cout << "Enter customer's phone number: ";

cin >> phone;

cout << "Enter airtime: ";

cin >> airtime;

cout<<"\n\n";

Subscriber subscriber(name, phone, airtime, bonus\_points); // Create an instance of the Subscriber class

subscriber.compute\_bonuspoints(); // Calculate the bonus points

subscriber.print\_info(); // Print the subscriber's info

return 0;

}

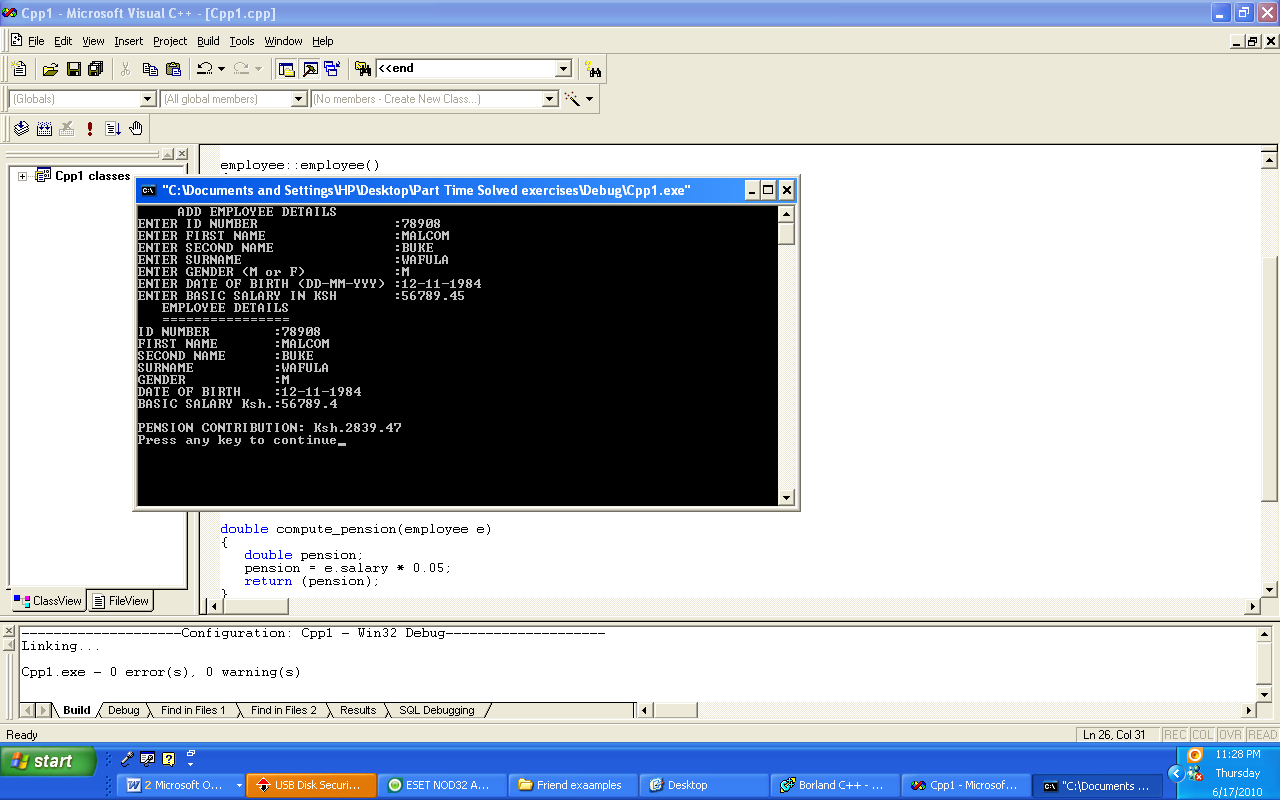
**OUTPUT**



**QUESTION 5 {10 Marks}**

The Finance Department of Fortune Business Systems Ltd has sought your help in developing a program that captures an employee’s details: employee ID, first name, second name, surname, gender, Date of Birth (entered in dd-mm-yyyy format) and monthly basic salary

1. (i) Create a class called **employee** that captures the above data using the class constructor. The class should also have a function **show\_employee(),** which should display employee information.
   1. Implement the class using an employee object called **emp\_obj.** Sample output is as below:



1. Employees of Fortune Business Systems Ltd contribute 5% of their monthly basic salary towards their individual pension savings. Since this may change in future, the operation to compute the pension has not been included. A friend function (to the class employee) called **compute\_pension()** to calculate and return the pension contribution of an employee is therefore needed. In addition to the output as shown in 3a(ii) above

**CODE**

# Define an Employee class

class Employee:

# Constructor to initialize Employee object with given attributes

def \_\_init\_\_(self, emp\_id, first\_name, second\_name, surname, gender, date\_of\_birth, basic\_salary):

self.emp\_id = emp\_id

self.first\_name = first\_name

self.second\_name = second\_name

self.surname = surname

self.gender = gender

self.date\_of\_birth = date\_of\_birth

self.basic\_salary = basic\_salary

# Method to display employee details

def show\_employee\_details(self):

print("\n\nEMPLOYEE DETAILS")

print("================")

print("Employee ID:", self.emp\_id)

print("Name:", self.first\_name, self.second\_name, self.surname)

print("Gender:", self.gender)

print("Date of Birth:", self.date\_of\_birth)

print("Monthly Basic Salary:", self.basic\_salary)

# Method to display employee pension details

def show\_employee\_pension(self):

print("\nPENSIONS")

print("===========")

print("Employee ID:", self.emp\_id)

print("Name:", self.first\_name, self.second\_name, self.surname)

print("Monthly Basic Salary:", self.basic\_salary)

print("Pension Contribution:", self.compute\_pension())

# Method to compute employee's pension contribution

def compute\_pension(self):

return 0.05 \* self.basic\_salary

# Create an empty list to store employee objects

employee\_list = []

# Create an instance of the Employee class and add it to the employee list

emp\_obj = Employee("78908", "MALCOM", "BUKE", "WAFULA", "M", "12-11-1984", 56789.45)

employee\_list.append(emp\_obj)

# Loop through the options and perform the corresponding actions based on user's choice

while True:

print("\nEMPLOYEE MANAGEMENT SYSTEM")

print("==========================")

print("1. Add new employee details")

print("2. Display employee details")

print("3. Display employee pensions")

print("4. Delete employee details")

print("5. Exit")

choice = input("Enter your choice: ")

# If user selects '1', add new employee details to the list

if choice == '1':

print("\nEnter the following employee details:")

emp\_id = input("Employee ID: ")

first\_name = input("First Name: ")

second\_name = input("Second Name: ")

surname = input("Last Name: ")

gender = input("Gender: ")

date\_of\_birth = input("Date of Birth (dd-mm-yyyy): ")

basic\_salary = float(input("Monthly Basic Salary: "))

employee = Employee(emp\_id, first\_name, second\_name, surname, gender, date\_of\_birth, basic\_salary)

employee\_list.append(employee)

print("\nEmployee details added successfully.")

# If user selects '2', display all employee details in the list

elif choice == '2':

if employee\_list:

for employee in employee\_list:

employee.show\_employee\_details()

else:

print("\nNo employee details found. Please add employee details first.")

# If user selects '3', display all employee pension details in the list

elif choice == '3':

if employee\_list:

for employee in employee\_list:

employee.show\_employee\_pension()

else:

print("\nNo employee details found. Please add employee details first.")

# If user selects '4', delete an employee record from the list based on the given employee ID

elif choice == '4':

emp\_id = input("Enter the ID of the employee you want to delete: ")

for employee in employee\_list:

if employee.emp\_id == emp\_id:

employee\_list.remove(employee)

print("\nEmployee record deleted successfully.")

break

else:

print("\nEmployee record not found.")

#if user selects '5' the program exits.

elif choice == '5':

print("\nExiting the program...")

break

else:

print("\nInvalid choice. Please select a valid option.")

**OUTPUTS**

