CONFIDENTIAL

SCHOOL OF COMPUTING
Faculty of Engineering

UNIVERSITI TEKNOLOGI MALAYSIA

FINAL EXAM – PROBLEM SOLVING
SET A

SEMESTER II 2019/2020

SUBJECT CODE	**: SECJ/SCSJ 1023**
SUBJECT NAME	**: PROGRAMMING TECHNIQUE II**
YEAR/COURSE	**: 1 (SECB/SECJ /SECP/SECR/SECV) 2 (SCSR/SCSV)**
TIME	**: 9.00 am – 11.00 am (2 Hours)**
DATE/ DAY	**: 9th JULY 2020 (THURSDAY)**

Answer Scheme

PROBLEM SOLVING (70 Marks)

Task 1: (2.5 Marks)

Audience class

```
//Task 1(a): Define the virtual displayDetails function [1.5 marks]
virtual void displayDetails() //0.5
{
    cout << left
         << setw(25) << name //0.5
         << setw(7)  << age; //0.5
}
```

```
//Task 1(b): Define the pure virtual calcDisc function [1 mark]
virtual double calcDisc() = 0; //pure virtual function
```

Task 2: (7.5 Marks)

Adults class

```
//Task 2(a): Specify the Adults class as a child of the Audience class
//[1 mark]
class Adults : public Audience //Inheritance - 1
```

```
//Task 2(b): Define the constructor with arguments [1.5 marks]
Adults (string n, int a, int s): Audience(n, a, s)//0.5 : //1
{}
```

```
//Task 2(c): Define the calcDisc function [2.5 marks]
double calcDisc() //0.5
{
    if (age < 60) //1
        return 0; //0.5
    else
        return DISC2; //0.5
}
```

```
//Task 2(d): Redefine the displayDetails function [2.5 marks]
void displayDetails() //0.5
{
    Audience::displayDetails(); //1
    cout << setw(25) << "-";  //0.5
    cout << setw(9)  << seatNo; //0.5
}
```

Task 3: (8 Marks)

Kids class

```
//Task 3(a): Specify the Kids class as a child of the Audience class [1
mark]
class Kids : public Audience //Inheritance - 1
```

```
//Task 3(b): Define the constructor with arguments [2 marks]
Kids (string n, int a, int s, string pn): Audience(n, a, s) //0.5 : //1
{
    parentName = pn; //0.5
}
```

```
//Task 3(c): Define the calcDisc function [2.5 marks]
double calcDisc() //0.5
{
    if (age > 2) //1
       return DISC2; //0.5
    else
       return DISC1; //0.5
}
```

```
//Task 3(d): Redefine the displayDetails function [2.5 marks]
void displayDetails() //0.5
{
    Audience::displayDetails(); //1
    cout << setw(25) << parentName; //0.5
    cout << setw(9)  << seatNo; //0.5
}
```

Task 4: (19.5 Marks)

Movie class

```
//Task 4(a): Define all the member variables [3.5 marks]
Date date; //(0.5) -> composition
Time time; //(0.5) -> composition
double price; //(2)
string title;
bool seat[N];
int hallNo, numAdults, numKids, numAudience;
Audience *audienceList[SIZE]; //(0.5) -> aggregation
```

```
//Task 4(b): Complete the definition of the readInput function [4 marks]
cout << "Title: ";
getline(cin, title); //(1)
cout << "Hall : ";
cin >> hallNo; //(0.5)
cout << "Ticket Price: RM";
cin >> price; //(0.5)
date.readDate(); //(1)
time.readTime(); //(1)
```

```
//Task 4(c): Complete the definition of the addAudience function
//[2.5 marks]
if (dynamic_cast <Adults*> (p))
{
    numAdults++; //(0.5)
}
```

```
else
{
   numKids++; //(0.5)
}
audienceList[numAudience] = p; //(1)
numAudience++; //(0.5)
```

```
//Task 4(d): Complete the definition of the displayInfoMovie function
//[3.5 marks]
cout << left;
cout << "Title: " << title  << endl //(0.5)
     << "Hall : " << hallNo << endl //(0.5)
     << "Date : ";
date.dispDate(); //(1)
cout << "\nTime : ";
time.dispTime(); //(1)
cout << fixed << setprecision(2)
      << "\nPrice: RM" << price << endl; //(0.5)
```

```
//Task 4(e): Complete the definition of the displayInfoAudience function
//[6 marks]
for (int i = 0; i < numAudience; i++) //(1)
{
   cout << (i+1) << ".  "; //(0.5)
   audienceList[i]->displayDetails(); //(1)
   priceAftDisc = price * (100 - audienceList[i]->calcDisc()) / 100.0; //(2)
   totalPrice += priceAftDisc; //(1)
   cout << priceAftDisc << endl; //(0.5)
}
```

Task 5: (25.5 Marks)

main function

```
//Task 5(a): Enter the task chosen [1 mark]
choice1 = menu(); //(1)
```

```
//In case 1:
//Task 5(b-i): Enter the details of movie [1 mark]
movieObj.readInput(); //(1)

//Task 5(b-ii): Add the Movie object into the Movie array [1 mark]
movieList[numMovie] = movieObj; //(1)
```

```
//In case 2:
//Task 5(c-i): Display the list of movies [3.5 marks]
for (int i = 0; i < numMovie; i++) //(1)
{
   cout << (i+1) << ") " << movieList[i].getTitle(); //(1.5) - 0.5 : 1
   movieList[i].getDateTime(); //(1)
   cout << endl;
}
```

```
//Task 5(c-ii): Check the seat availability. If the seat is available,
//update the status of the seat. [5 marks]
if (movieList[choice2-1].getSeat(seatNo-1)) //(2)
{
   movieList[choice2-1].setSeat(seatNo-1); //(2)
   status = false; //(1)
}

//Task 5(c-iii): Dynamically allocates a new audience (an adult
//or a kid) object. Use a polymorphic concept. [5 marks]
if (age > 12) //To identify the audience category
{
   audiencePtr = new Adults(name, age, seatNo); //(2)
}
else
{
   cout << "Parent Name: ";
   getline(cin, pName); //(1) -> Enter a parent name
   audiencePtr = new Kids(name, age, seatNo, pName); //(2)
}

//Task 5(c-iv): Add the audience to the selected Movie object [2 marks]
movieList[choice2-1].addAudience(audiencePtr); //(2)
```

```
//In case 3:
//Task 5(d): Display the list of movies' details [2.5 marks]
for (int i = 0; i < numMovie; i++) //(1)
{
    cout << "\nMovie #" << (i+1) << endl; //(0.5)
    movieList[i].displayInfoMovie(); //(1)
}
```

```
//In case 4:
//Task 5(e): Display the list of audiences' details based on the movie
//[3.5 marks]
for (int i = 0; i < numMovie; i++) //(1)
{
   cout << "\nMovie #" << (i+1) << endl; //(0.5)
   movieList[i].displayInfoMovie(); //(1)
   movieList[i].displayInfoAudience(); //(1)
}
```

```
//Task 5(f): Enter the task chosen [1 mark]
choice1 = menu(); //(1)
```

Task 6: (7 Marks)

```
//(a) The program is able to run (4 marks)
/*- Compile with error (1)
  - Run with error (2)
  - Run with inaccurate output and/or inproper display (3)
  - Run with accurate output and proper display (4)*/
```

```
//(b) Using an appropriate structure for the program (3 marks)
/*- Proper output formatting used (2)
  - Proper indentation and C++ statements used (1)*/
```