**Challenge- 1:** *Time Calculator* **(**5**)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <ctime>

using namespace std;

struct Time

{

int hours;

int minutes;

int seconds;

};

Time getTimeDifference(Time rfs)

{

Time currentTime;

time\_t t = time(0);

tm curTime = \*localtime(&t);

currentTime.hours = curTime.tm\_hour;

currentTime.minutes = curTime.tm\_min;

currentTime.seconds = curTime.tm\_sec;

long long int currentTotalSec = currentTime.hours \* 3600 + currentTime.minutes \* 60 + currentTime.seconds;

long long int rfsTotalSec = rfs.hours \* 3600 + rfs.minutes \* 60 + rfs.seconds;

long long int timeDifferenceInSec = currentTotalSec - rfsTotalSec;

timeDifferenceInSec < 0 ? timeDifferenceInSec \*= -1 : timeDifferenceInSec;

Time timeDifference;

timeDifference.hours = timeDifferenceInSec / 3600;

timeDifference.minutes = (timeDifferenceInSec % 3600) / 60;

timeDifference.seconds = timeDifferenceInSec % 60;

return timeDifference;

}

int main()

{

Time t = { 10, 15, 50 };

Time diffTime = getTimeDifference(t);

cout << diffTime.hours << " : " << diffTime.minutes << " : " << diffTime.seconds;

cout << endl;

return 0;

}

**Sample Runs:**

**21:10:30 – 20:30:55 = 0:39:35 2**

**12:10:50 – 6:6:55 = 6:3:55 0.25**

**15:15:15 – 15:15:15 = 0:0:0 0.25**

**21:30:5 – 21:29:55 = 0:0:10 2**

**0:0:0 - 0:0:30 = 0:0:30 0.5**

**Challenge- 2:** *Token Machine & Service Counters* **(**21.5**)**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <ctime>

using namespace std;

struct Time

{

int hours;

int minutes;

int seconds;

};

struct Date

{

int day;

int month;

int year;

};

struct Token

{

int tokenNumber;

Date tokenDate;

Time tokenTime;

};

struct ServiceCounter

{

int counterNumber;

Token list[100];

int tokensCount;

};

void printDate(Date d) **(0.5)**

{

cout << "Date: " << d.day << ":" << d.month << ":" << d.year;

}

void printTime(Time t) **(3)**

{

char bound = 'A';

if (t.hours > 12)

{

t.hours = t.hours - 12;

bound = 'P';

}

else if (t.hours == 12)

{

bound = 'P';

}

else if (t.hours == 0)

{

t.hours = 12;

}

cout << "Time: " << t.hours’ << ":" << t.minutes << ":" << t.seconds << " " << bound << 'M';

}

**Sample Runs:**

**23:59:45 => 11:59:45 PM 0.5**

**14:30:15 => 2:30:15 PM 0.5**

**00:00:00 => 12:00:00 AM 1**

**12:00:00 => 12:00:00 PM 1**

Time getCurrentTime()

{

time\_t t = time(0);

tm curTime = \*localtime(&t);

currentTime.hours = curTime.tm\_hour;

currentTime.minutes = curTime.tm\_min;

currentTime.seconds = curTime.tm\_sec;

}

Date getCurrentDate()

{

currentDate.day = curTime.tm\_mday;

currentDate.month = curTime.tm\_mon + 1;

currentDate.year = curTime.tm\_year + 1900;

}

Token generateToken() **(3.5)**

{

static int tokenCounter = 1; **3**

Time currentTime = getCurrentTime();

Date currentDate = getCurrentDate();

Token newToken;

newToken.tokenNumber = tokenCounter;

tokenCounter++;

newToken.tokenDate = currentDate;

newToken.tokenTime = currentTime;

return newToken;

}

**Making counter local static : 2.5**

**If made global : 1**

**Remaining logic : 1**

void printToken(Token t) **(1.5)**

{

cout << "\nToken # : " << t.tokenNumber;

cout << '\n';

printTime(t.tokenTime); **0.5**

cout << '\n';

printDate(t.tokenDate); **0.5**

}

**Atomicity to print Date and Time : 1**

**Remaining logic : 0.5**

void initializeServiceCounter(ServiceCounter& s) **(3)**

{

static int tokenGenerator = 1; **3**

s.counterNumber = tokenGenerator;

tokenGenerator++;

s.tokensCount = 0;

}

**Making counter local static : 2.5**

**If made global : 1**

**Remaining logic : 0.5**

void assignToken(ServiceCounter& s, Token t) **(3)**

{

s.list[s.tokensCount] = t;

s.tokensCount++;

}

void serviceToken(ServiceCounter& s) **(4)**

{

if (s.tokensCount == 0) **1**

return;

for (int i = 0; i < s.tokensCount-1; i++) **2**

{

s.list[i] = s.list[i + 1];

}

s.tokensCount--; **1**

}

**Checking if tokensCount is 0 : 1**

**Remaining logic to remove token : 3**

void printTokensAtServiceCounter(ServiceCounter s) **(3)**

{

cout << "\n\*\*\* Token Queue Waiting For Service @ Counter # " << s.counterNumber;

if (s.tokensCount == 0) **1**

{

cout << "\n\t\t --> Queue is Empty <--";

return;

}

for (int i = 0; i < s.tokensCount; i++)

{

cout << '\n';

printToken(s.list[i]);

}

}

**Checking if tokensCount is 0 : 1**

**According to sample output : 1**

**Remaining logic : 1**

int main()

{

ServiceCounter a, b;

initializeServiceCounter(a);

initializeServiceCounter(b);

assignToken(a, generateToken());

assignToken(b, generateToken());

assignToken(a, generateToken());

assignToken(a, generateToken());

assignToken(b, generateToken());

assignToken(b, generateToken());

assignToken(b, generateToken());

assignToken(b, generateToken());

assignToken(b, generateToken());

assignToken(a, generateToken());

printTokensAtServiceCounter(a);

serviceToken(a);

printTokensAtServiceCounter(a);

serviceToken(a);

serviceToken(a);

serviceToken(a);

printTokensAtServiceCounter(a);

serviceToken(b);

printTokensAtServiceCounter(b);

return 0;

}

**Question 1:**

**Sample Runs:**

**21:10:30 – 20:30:55 = 0:39:35 2**

**12:10:50 – 6:6:55 = 6:3:55 0.25**

**15:15:15 – 15:15:15 = 0:0:0 0.25**

**21:30:5 – 21:29:55 = 0:0:10 2**

**0:0:0 - 0:0:30 = 0:0:30 0.5**

**Question 2:**

printDate **0.5**

printTime

**Sample Runs:**

**23:59:45 => 11:59:45 PM 0.5**

**14:30:15 => 2:30:15 PM 0.5**

**00:00:00 => 12:00:00 AM 1**

**12:00:00 => 12:00:00 PM 1**

generateToken

**Making counter local static : 2.5 → If made global : 1**

**Remaining logic : 1**

**(total 3.5)**

printToken

**Atomicity to print Date and Time : 1**

**Remaining logic : 0.5**

initializeServiceCounter

**Making counter local static : 2.5 → If made global : 1**

**Remaining logic : 0.5**

assignToken **3**

serviceToken

**Checking if tokensCount is 0 : 1**

**Remaining logic to remove token : 3**

printTokensAtServiceCounter

**Checking if tokensCount is 0 : 1**

**According to sample output : 1**

**Remaining logic : 1**

**Atomicity → - 1 marks from overall.**

**Penalty Matrix:**

| Penalty List | Labs | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Indentation putting { Infront of loop header, in do while, putting while with closing } | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Meaningful Variable Names |  | -2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Camel Case Notation | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Atomicity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Syntax error | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Linker error | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Wrong function prototypes | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Class interface or additional members |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Use of library function/class without permission | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Continue statement | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| cin/cout where it isn’t needed | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Multi-filing |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Wrong #ifndef or name of header file |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Global functions |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Multiple classes in one header file |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |