

8.0

University of Eswatini

Department of Computer Science

Resit Examination: January 2020

Title of paper : Computer Programming II

Course Number : CSC213

Time Allowed : Three (3) hours

This paper may not be opened until permission has been granted by the invigilator

INSTRUCTIONS

1. Answer all questions.
2. Examination has a total of 80 marks
3. This exam consists of 11 printed pages including the cover page.
4. The Exam user_id, password, tree, context and server name will be provided by the chief invigilator.
5. Read the complete question paper carefully before starting to work on the problem.
6. Write pseudo codes (hand-written) in the provided answer folder.
7. Submit written answer folder and zipped project folder
8. Use the last 10 minutes to check your submissions
9. The names of all your files(project, source file and output files) should have following format

```
S-----(Project Name)
S-----.cpp (source file)
S-----.TXT (data files)
```

The dashes in file names are the six digits of your UNESWA student identity number.

SPECIAL REQUIREMENTS:

1. For each student, a standalone PC with working Visual Studio 2010 C++ compiler.
2. Students **should not** have access to the internet.

ANSWER FORMAT

1. Where required, write (in your answer folder) a detailed pseudo-code.
2. Compile and test your code. Make sure you submit code with no syntax errors. Where necessary comment statements that have syntax errors.
3. Provide sufficient comment in your source code.
4. Output from your program must be properly formatted.

DATA

1. The required data text files, and ANNEX_A source files, are stored in the folder RESIT2020_CSC213_DATA_ANNEX and will be provided by the chief invigilator.
2. Except where instructed, the data files and the ANNEX source files should not be modified. However, where necessary content can be used in your program.

PROBLEM:

Your task is to design and implement a program that reads student information stored in three comma separate (.csv) files, and extract a report for each student. The content of the files is as follows:

- **students.csv** – This file contains 25 unique student records consisting of the student identity number, surname and first name.
- **courses.csv** – This file contains 37 unique course records. Each record is described by course code, course title and the weighting of the continuous assessment (CA) and examination marks. The weighting is a value between 0 and 1, and the sum of CA and examination weighting must be equal to 1.
- **enrolments.csv** – Each student can register in one or more courses, and each course can be taken by many students. This file contains a record of each student identity number and the course codes for the courses they are registered in. For each registration, the CA and examination marks are also recorded. The file contains 148 unique enrolment records.

The figure below shows a sample of each of the three files. For testing purposes, electronic copies of the input files are provided. For testing purposes, electronic copies of the input files are provided.

students.csv				courses.csv			
StudentId	Surname	Name		CourseId	StudentId	CAMark	examMark
604818	SMITH	JAMES		CSC111	604818	35	52
542961	JOHNSON	JOHN		CSC211	542961	4	61
778736	WILLIAMS	ROBERT		CSC242	778736	82	63
952551	BROWN	MICHAEL		CSC341	952551	3	63
340674	JONES	WILLIAM		CSC111	340674	40	1
320851	GARCIA	DAVID		CSC211	320851	94	91
587820	MILLER	RICHARD		CSC242	587820	83	54
935489	DAVIS	JOSEPH		CSC272	935489	53	1
663665	RODRIGUEZ	THOMAS		CSC111	663665	3	16

83

CODE	DESCRIPTION	CA_RATE	EXAM_RATE
CSC111	Introduction to Computer Science	0.4	0.6
CSC113	Introduction to Information Technology	0.4	0.6
CSC121	Communications Fundamentals	0.4	0.6
CSC112	Computer Programming I	0.4	0.6
CSC203	Discrete Mathematics	0.4	0.6
CSC205	Probability and Statistics	0.4	0.6
CSC213	Computer Programming II	0.4	0.6
CSC251	Human Computer Interaction	0.4	0.6

The program must repeatedly read each student record in the students.csv file and extracts all corresponding/matching enrolment records (matched using unique student identity number) from the enrolments.csv file. The course code in the enrolment record is also used to extract and display matching records (matched using unique course code) in the courses.csv file. For each student, the program should produce a student report similar to figure shown below:

```

C:\Users\Lukhetso\2\Desktop\TO COPY TO OFFICE MACHINE\July209_SampleSol_VA\Debug\July209_S
Student Identity Number: 979144
Student Surname : THOMPSON
Student Firstname : SUSAN
CID TITLE CG EXAM PINRL GRADE
-----
CSC242 Object Oriented Programming 69 78 74 B
CSC437 Computer Networks I 8 72 46 E
CSC111 Introduction to Computer Science 83 37 55 D
CSC213 Computer Programming II 89 48 64 C
CSC251 Human Computer Interaction 57 89 76 B
CSC304 Internship Training 99 79 87 A
OVERALL AVERAGE 68 67 67 C

```

QUESTION 1 – 15 marks

Create a new Visual Studio C++ project and add the code provided in ANNEX A to the main source file. Where indicated in ANNEX A, add code for the following:

- (a) Write a function that's determines a letter grade for given a mark using the following integer ranges. A: 80-100; B: 70-80; C: 60-70; D: 50-60; E: 40-50; F: 0-40. The function takes a single numeric value and returns letter grade. The function must adequately resolve border line cases. [5 marks]
- (b) Write a function whose arguments are a given CA mark, examination mark, CA weighting and examination weighting and computes the final mark as a sum of the product of each mark (CA or Exam) and its corresponding weighting. Final = CA mark * CA weight + Exam Mark* Exam weight. [5 marks]
- (c) Write a function whose arguments are a character, an integer number N, and an output stream. The function writes the given character N times to output stream. [5 marks]

QUESTION 2 – 25 marks

48

Where indicated in the code provided in ANNEX A, modify the code as follows.

- (a) Define a course record structure, called **struct Course**, to store information about each course record. That is the structure must have fields for the course code, course title, CA weighting and examination weighting. Make sure your variable names are consistent with those used in the given code from ANNEX A. [4 marks]
- (b) Define a function called **InitCourseRecord** that initializes a course record. The function takes as an argument, a course record (as described above), and values for each field. The function simple sets the record fields to the given values. [3 marks]
- (c) Write an overload insertion operator for that takes a struct Course variable and displays the course details in the following format.

```
GSC213 Programming II 0.4 0.6
```

- (d) In your main function, add statements to test the **InitCourseRecord** function and the insertion operator.
- (e) In the main function, run and test the **extractCourseDataFromCSVFile** and **displayCourseTable** [The definitions of these two functions is provided in the Annex code] and make sure it displays a list of courses on standard output similar to figure below. Note: *This will only work if parts (a) and (b) above are correctly implemented.* [3 marks]

```
Sample Testing Code
//extract course record from csv file to list
std::list<Course> CT = extractCourseDataFromCSVFile("courses.csv");
//display course list
displayCourseTable(CT, std::cout);
```

Expected result

No.	CODE	DESCRIPTION	COURSE	EXAMRATE
1	GSC111	Introduction to Computer Science	0.40	0.60
2	GSC113	Introduction to Information Technology	0.40	0.60
3	GSC121	Communications Fundamentals	0.40	0.60
4	GSC112	Computer Programming I	0.40	0.60
5	GSC203	Discrete Mathematics	0.40	0.60
6	GSC205	Probability and Statistics	0.40	0.60
7	GSC213	Computer Programming II	0.40	0.60

69

(f) In the **displayCourseTable** function, replace the statement that writes a line of equal character symbol (=): `os << "====="` << `std::endl`; with a statement that calls the function defined in question 1(c). [3 marks]

(g) Write a function that take a course code and a course table (list of courses extracted using the **extractCourseDataFromFile** function) and returns the complete course record which matches the course code passed as an argument. Here is the suggested prototype for the function. [12 marks]

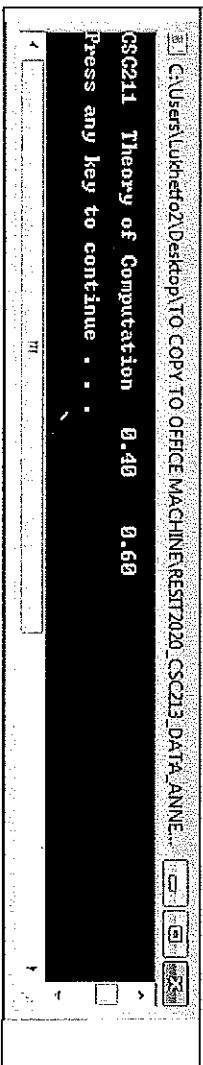
Course getCourseRecordFromList(std::string, std::list<Course>)

Here is a sample of how the function could be used and tested in the main function

```
Sample: Testing Code

//get course record from lis
std::list<Course> CT = extractCourseDataFromFile("courses.csv");
Course CX = getCourseRecordFromList("CSC211", CT);
std::cout << CX.code << "\t" << CX.description << "\t"
          << CX.cakate << "\t" << CX.examRate << "\t" << std::endl;

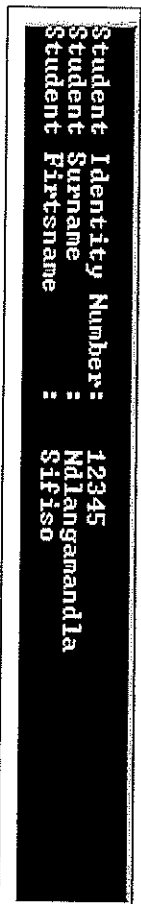
Expected result
```



QUESTION 3 – 40 marks

Where indicated in the code provided in ANNEX, modify the code as follows.

- (a) Define a student record structure, **called struct Student**, to store information about each student record. That is the structure must have fields for the student identity number, surname and first name. [3 marks]
- (b) Write an overloaded insertion operator for struct Student which writes the student identity number, surname and first name to a stream in the format shown below:



```
Student Identity Number: 12345
Student Surname      : Ndlangamandla
Student Firstname    : Sifiso
```

[5 marks]

Test your code as shown below:

Sample Testing Code	Expected result
<pre>//declare student record Student S; //initialize student record S.sid = "12345"; S.surname = "Ndlangamandla"; S.firstname = "Sifiso"; //display on standard output using insertion operator std::cout << S << std::endl;</pre>	<pre>Student Identity Number: 12345 Student Surname : Ndlangamandla Student Firstname : Sifiso Press any key to continue . . .</pre>

- (c) Write a function that repeatedly reads each student record in the students.csv file and extracts all corresponding/matching enrolment records (matched using unique student identity number) from the enrolments.csv file. The provided ANNEX A code provides enough samples for how to write the code for this function. Here is a suggested prototype for the function:

```
void extractStudentResults (const char* studentFilename, const char*
enrolmentFilename, std::ostream& os)
```

 [25 marks]
For each student, the function should produce a student report similar to the figure below:

```

C:\Users\Lukhetro2\Desktop\TO_COPY_TO_OFFICE_MACHINE\July20...
Student Identity Number: 979144
Student Surname : THOMPSON
Student Firstname : SUSAN

CID SID CR EXAM FINL GRADE
-----
CSG242 979144 69 78 74 B
CSG437 979144 8 72 46 D
CSG111 979144 83 37 55 C
CSG213 979144 89 48 64 C
CSG251 979144 57 89 76 B
CSG304 979144 99 79 87 R
OVERALL AVERAGE 68 67 67 C

```

(d) As noted in the output above, the student identity number column in the report is redundant. Explain how the code could be modified to display the course title/description instead as shown below. Write the pseudo-code for the modified function. [7 marks]

```

C:\Users\Lukhetro2\Desktop\TO_COPY_TO_OFFICE_MACHINE\July209_SampleSol_1\Debug\July209_S...
Student Identity Number: 979144
Student Surname : THOMPSON
Student Firstname : SUSAN

CID TITLE CR EXAM FINL GRADE
-----
CSG242 Object Oriented Programming 69 78 74 B
CSG437 Computer Networks I 8 72 46 D
CSG111 Introduction to Computer Science 83 37 55 C
CSG213 Computer Programming II 89 48 64 C
CSG251 Human Computer Interaction 57 89 76 B
CSG304 Internship Training 99 79 87 R
OVERALL AVERAGE 68 67 67 C

```


ANNEX A:

(This code is provided in the data folder: RESIT2020_CSC213_DATA_ANNEX_A)

```
//ANNEX
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>

//include list STL
#include <list>
#include <iterator>

//work with string streams
#include <sstream>

/*****
*/
CONSTANT DECLARATIONS
/*****
*/
//constant declarations used in std::setw() statements
const int SID_W = 12;
const int CA_RATE_W = 12;
const int CA_W = 7;
const int CID_W = 10;
const int EXAM_RATE_W = 12;
const int EXAM_W = 7;
const int FINAL_W = 7;
const int GRADE_W = 7;
const int SEQ_W = 5;
const int SURNAME_W = 15;
const int FIRSTNAME_W = 20;
const int TITLE_W = 40;
/*****
*/
END CONSTANT DECLARATIONS *****/

/*****
*/
QUESTION 1
/*****
*/
//Q1-a
//ADD FUNCTION DEFINITION HERE
// determine final mark grade

//Q1-b
//ADD FUNCTION DEFINITION HERE
//calculate final mark

//Q1-c
//ADD FUNCTION DEFINITION HERE
//write a character n times to an output stream
/*****
*/
END QUESTION 1 CODE *****/

/*****
*/
QUESTION 2
/*****
*/
//Q2-a
//ADD YOUR CODE HERE

//Q2-b
//ADD YOUR CODE HERE

//Q2-c
//ADD YOUR CODE HERE
```

92

93

```

/*****
/extract course data from csv file
std::list<Course> extractCourseDataFromCSVFile (const char* courseFilename)
{
    std::list<Course> courseTable;

    //open input file for reading
    std::ifstream inf;
    inf.open(courseFilename, std::ios::in);
    std::string lineStr="";

    Course courseRec;

    //ignore input file header text
    getline(inf, lineStr);

    //Loop to extract each line
    while(getline(inf, lineStr))
    {
        //To read lines into string stream
        std::stringstream lineSS(lineStr);

        //declare strings for code,description, carate, examrate
        std::string code_val, descr;
        std::string ca_rate, exam_rate;

        //read data from file
        //lineSS >> code_val >> descr >> ca_rate >> exam_rate ;
        getline(lineSS, code_val, ',');
        getline(lineSS, descr, ',');

        getline(lineSS, ca_rate, ',');
        //convert string to double
        double carate_val = std::strtod(ca_rate.c_str(),NULL);

        getline(lineSS, exam_rate, ',');
        //convert string to double
        double examRate_val = std::strtod(exam_rate.c_str(),NULL);

        //init
        InitCourseRecord(courseRec, code_val, descr, carate_val, examRate_val);

        courseTable.push_back(courseRec);
    }
    inf.close(); // close input file
    return courseTable; // return number frequency table
}

//display course table
void displayCourseTable(std::list<Course> courseTable, std::ostream& os)
{
    // write header and values to an output stream
    os << "===== " << std::endl<< std::endl;
    os << std::left << std::setw(SEQ_W) << "No. ";
    os << std::left<<std::setw(CID_W)<<"CODE"
        <<std::setw(TITLE_W)<<"DESCRIPTION"
        <<std::setw(CA_RATE_W)<<"CARATE"
        <<std::setw(EXAM_RATE_W)<<"EXAMRATE"
        << std::endl<< std::endl;
    os << "===== " << std::endl<< std::endl;
    int courseCount = 0;
    for(std::list<Course>::iterator it = courseTable.begin();
        it != courseTable.end(); it++)
    {
        courseCount += 1;
        std::stringstream ss;

        ss << courseCount;
        std::string courseCount_str = ss.str();
    }
}

```

```

os<< std::left << std::setw(CID_W) << it->code
os<<std::setw(CID_W)<< it->code
<<std::setw(TITLE_W)<< it->description
<< std::fixed << std::setw(CA_RATE_W)<< std::setprecision(2) << it->caRate
<< std::fixed << std::setw(EXAM_RATE_W)<< std::setprecision(2)<< it->examRate
<< std::endl<< std::endl;
}
os << "===== " << std::endl<< std::endl;
}

/***** END QUESTION 2 CODE *****/
/*****
*/
QUESTION 3
/*****
*/
//Q3-a
// ADD YOUR CODE HERE

//Q3-b
// ADD YOUR CODE HERE

//Q3-c
// ADD YOUR CODE HERE
/***** END QUESTION 3 CODE *****/
int _tmain(int argc, _TCHAR* argv[])
{
//TEST CODE
// ADD YOUR CODE
system ("pause");
return 0;
}

```