

Tutorial letter 101/3/2017

Introduction to Programming II COS1512

Semesters 1 & 2

School of Computing

IMPORTANT INFORMATION:

Please activate your *myUnisa* and *myLife* email addresses and ensure you have regular access to the *myUnisa* module site COS1512-2017-S1 or COS1512-2017-S2 as well as your e-tutor group site.

Note: This is a blended online module, and therefore your module is available on myUnisa. However, in order to support you in your learning process, you will also receive some study materials in printed format. Please visit the COS1512 course website on myUnisa at least twice a week.

CONTENTS

	<i>Page</i>
1 INTRODUCTION AND WELCOME	4
2 OVERVIEW OF THE MODULE COS1512.....	5
2.1 Purpose	5
2.2 Outcomes	6
3 LECTURER(S) AND CONTACT DETAILS.....	7
3.1 Lecturer(s)	7
3.2 Department.....	7
3.3 University.....	8
4 MODULE-RELATED RESOURCES.....	8
4.1 Joining myUnisa	8
4.2 Other resources – Printed support materials.....	9
4.3 Prescribed books	9
4.4 Recommended books	10
4.5 Prescribed software	10
4.6 myUnisa Forum	10
4.7 Tutorial letters	10
4.8 Additional Resources	10
5 STUDENT SUPPORT SERVICES FOR THE MODULE	11
5.1 E-Tutors.....	11
5.2 Free computer and internet access.....	11
5.3 Downloading study material and software.....	12
5.4 Additional Resources on myUnisa	12
5.5 Announcements on myUnisa	12
5.6 Installation of the software	12
6 MODULE-SPECIFIC STUDY PLAN	12
6.1 Syllabus	12
6.2 Planning your academic year	13
6.3 Hints on studying this module	15
7 MODULE PRACTICAL WORK AND WORK-INTEGRATED LEARNING	16
8 ASSESSMENT	16
8.1 Assessment plan	16
8.2 General assignment numbers.....	19

8.2.1	Unique assignment numbers	19
8.2.2	Due dates for assignments	19
8.3	Submission of assignments	19
8.4	Assignments	21
9	EXAMINATION.....	54
10	FREQUENTLY ASKED QUESTIONS	54
11	CONCLUSION	55
12	APPENDIX A: THE SOFTWARE FOR COS1512.....	55
13	APPENDIX B: SOURCE LISTINGS OF THE SAMPLE PROGRAMS IN THE TEXTBOOK.....	62
14	APPENDIX C: GLOSSARY.....	62
15	APPENDIX D: Solution to assignment 3.....	63

Please note / important note: COS1512 is a semester module. You need AT LEAST eight hours per week for this module.

If you do not receive your study material immediately after registration, you have to download it from myUnisa so that you are able to start IMMEDIATELY with your studies. See section 5.2 in this tutorial letter for details about the downloading of study material.

To gain admission to the examination you have to submit Assignment 1 in time. The due date is 23 February if you are registered for the first semester and 11 August if you are registered for the second semester.

The COSALLF/301/0/2016 tutorial letter contains important general information that you will need during the year such as the names and contact details of lecturers assigned to the different modules.

1 INTRODUCTION AND WELCOME

Dear Student

Welcome to COS1512. We hope that you will find this module interesting and stimulating and that you will increase your knowledge about and your skills in programming in C++. We shall do our best to make your study of this module successful. In order to succeed with your studies, you need to start studying immediately and do the assignments properly.

This Tutorial Letter 101 contains important information about the scheme of work, resources and assignments for this module. We urge you to read it carefully and to keep it at hand when working through the study material, preparing the assignments, preparing for the examination and addressing questions to your lecturers.

Please read Tutorial Letter 301 and the *myStudies@Unisa* brochure in combination with Tutorial Letter 101 as it gives you an idea of generally important information when studying at a distance university and within a particular College.

In Tutorial Letter 101, you will find the assignments and assessment criteria as well as instructions on the preparation and submission of the assignments. This tutorial letter also provides all the information you need with regard to the prescribed study material and other resources and how to obtain it. Please study this information carefully and make sure that you obtain the prescribed material as soon as possible.

We have also included certain general and administrative information about this module. Please study this section of the tutorial letter carefully.

Because this is a blended online module, you need to use myUnisa to study and complete the learning activities for this course. You need to visit the website on myUnisa for COS1512 frequently. The website for COS1512 for the **first semester** is **COS1512-17-S1** and for the **second semester** it is **COS1512-17-S2**.

We hope that you will enjoy this module and wish you all the best!

1.1 To get started...

Because this is a blended online module, you need to go online to see your study materials and read what to do for the module. Go to the myUnisa website here: <https://my.unisa.ac.za> and login with your student number and password. You will see **COS1512-17-S1** (for the first semester) or **COS1512-17-S2** (for the second semester) in the row of modules in the orange blocks across the top of the webpage. Remember to also check in the **-more-** tab if you cannot find it in the orange blocks. Click on the module you want to open.

In addition, you will receive this tutorial letter and a printed copy of the online study materials from your module. While these printed materials may appear to be different from the online study materials, they are exactly the same and have been copied from the online myUnisa website.

1.1.1 About myUnisa

myUnisa is the student website that allows you to connect with your lecturers, e-tutors and fellow students, download your study material, submit assignments, gain access to the Library and various learning resources and participate in online discussion forums.

We also use myUnisa for announcements, and to deliver additional study material. Please join myUnisa and visit the COS1512 course website regularly.

1.2 Tutorial Matter

The tutorial matter for this module consists of the following:

- this tutorial letter, COS1512/101/3/2017;
- tutorial letter COSALLF/301/4/2017;
- a CD containing the prescribed C++ software (Disk2017);
- additional tutorial letters, containing additional information or solutions to assignments.

When you register, you will receive an **inventory letter** containing information about your tutorial matter. See also the brochure entitled *myStudies@Unisa*, (which you received with your tutorial matter). Check the study material that you have received against the inventory letter. You should have received all the items listed in the inventory, unless there is a statement like “out of stock” or “not available”. If any item is missing, follow the instructions on the back of the inventory letter without delay.

Some of this study material may not have been available when you registered. Study material that was not available when you registered will be posted to you as soon as possible, but is also available on myUnisa.

Please do not contact the School about missing tutorial matter, cancellation of a module, payments, enquiries about the registration of assignments, and so on, but rather the relevant department as indicated in the *myStudies@Unisa* brochure. The School should only be contacted about academic matters.

If you have not received all of the above mentioned tutorial matter, please contact our DESPATCH DEPARTMENT, using the contact details as given in the *My Studies @ Unisa* brochure. In the meantime, please download the study material from myUnisa.

All information that is made available electronically in the form of tutorial letters will also be sent to you on paper (except in a very few rare cases). Note that the determining information comes from the paper documents that you receive. If there is any conflict re for example, dates, prices, times, due dates, assignment numbers, et cetera, refer to the information published on paper or announcements on myUnisa, unless otherwise instructed.

2 OVERVIEW OF THE MODULE COS1512

2.1 Purpose

COS1512 is one of a number of first-year Computer Science modules offered by the School of Computing at Unisa.

COS1512 focuses on providing an introduction to objects and the object-oriented programming environment using C++ as programming language. The following topics are covered:

- file I/O streams as an introduction to objects and classes;
- using pre-defined classes such as string and vector;
- C strings, pointers and dynamic arrays;
- ADTs (i.e. user-defined classes including the functions and operators for these classes as well as separate compilation);
- recursion;
- single inheritance, and
- function and class templates.

The paragraphs below show where COS1512 fits into the programming modules offered by the School of Computing:

COS1511 deals with the basic concepts of programming, using the programming language C++. It is aimed at students who have not done any programming before. It is a pre-requisite for COS1512.

COS1512 introduces the learner to objects and the object-oriented programming environment.

COS1521 provides a general background to computer systems.

INF1511 is an introductory course in Delphi programming.

COS1501 introduces the mathematics relevant to Computer Science.

2.2 Outcomes

Once you have completed this module, you should have reached the following outcomes:

Outcome 1:

You should be able to design a logical solution to a simple programming problem, making appropriate assumptions.

Assessment criteria:

- Through assignments, including multiple choice and written assignments and an examination at the end of the semester, you are assessed on your ability to:
- interpret a problem description which specifies the requirements of a program;
- identify all steps necessary to solve a problem and order the steps in the correct logical sequence;
- write down the logical sequence of operations that a computer should perform to solve a particular problem;
- apply object-oriented principles during problem solving.

Outcome 2:

You should be able to write C++ program code, demonstrating the principles of good programming style.

Assessment criteria:

Through assignments, including multiple choice and written assignments and an examination at the end of the semester, you are assessed on your ability to:

- use the different C++ programming constructs appropriately and correctly, in order to implement a solution to a programming problem;
- write functions and use them in a program;
- define classes and use object-oriented principles to implement programming problems;
- recognise/locate errors in a program and correct them.

Outcome 3:

You should be able to demonstrate an understanding of the theory underlying the basic programming concepts.

Assessment criteria:

Through assignments, including multiple choice and written assignments and an examination at the end of the semester, you are assessed on your ability to:

- explain the purpose of a particular C++ programming construct and identify problem descriptions where they are applicable;
- define relevant programming concepts.

The specific learning *objectives* for each chapter in the prescribed book for COS1512 in order to reach the above learning outcomes are given in more detail in the study guide included in Appendix D of this tutorial letter.

3 LECTURER(S) AND CONTACT DETAILS

3.1 Lecturer(s)

If you experience problems with this subject or have any other enquiry about it, please feel free to contact the lecturers. The names and telephone numbers of your lecturers for this module, as well as the module e-mail address you can use for any queries regarding this module, are supplied in tutorial letter COSALLF/301/0/2017.

Email: Email is a convenient and the most effective way of communicating with a lecturer. **Students registered for the first semester should send e-mail queries to COS1512-17-S1@unisa.ac.za**

and students registered for the second semester should send e-mail queries to COS1512-17-S2@unisa.ac.za.

Do not email the lecturer directly - rather use the module email address, as a specific lecturer may not be available. Using the module email ensures someone will read it. Include the module code and your student number in the subject header of the message. Ask specific questions. It is difficult to respond properly via email to a request such as 'I don't understand problem 5.1. Please explain.' Always state exactly what it is that you do not understand.

Phone us: You are welcome to phone us directly, but please consult your tutorial letters and the myUnisa discussion forum to ensure that your query has not already been addressed. We are sometimes unavailable due to other departmental, research or university duties. If you fail to reach us directly, please phone the secretary. In urgent cases you may also leave a message at a secretary for us to call you back. The best contact hours are between 9:00 and 13:00 in the mornings, except Wednesdays when we have our departmental meetings.

3.2 Department

Please note that the School of Computing has moved to Florida in 2013. The School of Computing can be contacted telephonically at 011 670 9200 or via e-mail with the e-mail address computing@unisa.ac.za. Should you be unable to reach any of the lecturers for COS1512, please leave a message and your contact details with one of the secretaries, who can be contacted via the number given above. Remember to include the module code and your student number with the message.

3.3 University

Lecturers for this module are only responsible for content-related queries about the study material used for COS1512.

If you need to contact the University about matters not related to the content of this module, please consult the publication **myStudies@Unisa** that you received with your study material. This brochure contains information on how to contact the University (e.g. to whom you can write for different queries, important telephone and fax numbers, addresses and details of the times certain facilities are open).

NB: Always have your student number at hand when you contact the University.

4 MODULE-RELATED RESOURCES

4.1 Joining myUnisa

If you have access to a computer that is linked to the internet, you can quickly access resources and information at the University. The *myUnisa* learning management system is Unisa's online campus that will help you to communicate with your lecturers, with other students and with the administrative departments of Unisa – all through the computer and the internet.

You can start at the main Unisa website, <http://www.unisa.ac.za>, and then click on the *myUnisa* orange block. This will take you to the *myUnisa* website. To go to the *myUnisa* website directly, go to <https://my.unisa.ac.za>. When you are on the *myUnisa* website, click on the “Claim UNISA Login” at the right-hand side of the screen. You will then be prompted to give your student number to claim your initial myUnisa as well as myLife login details.

This module is presented following a blended approach in the sense that even though you will still receive some printed study material, most of the information needed to complete this module is available on *myUnisa*. *myUnisa* has inter alia the following tools which you will use regularly:

<i>Assignments</i>	This is a tool to manage your assignments; to submit, track and see marks obtained.
<i>Official Study Material</i>	All official study material are available under Official Study Material
<i>Additional Resources</i>	The Additional Resources tool contains a variety of resources related to COS1512. For example the Code::Blocks software.
<i>Course Contact</i>	The Course Contact tool facilitates e-mail communication between students and lecturers. Students use their myLife e-mail to send e-mails to their lecturers through myUnisa.
<i>Frequently Asked Questions</i>	Frequently asked questions, or FAQs, are listed questions and answers, all supposed to be frequently asked in some context, and pertaining to a particular topic.
<i>Learning Units</i>	The Learning Units tool contains learning units/lessons for COS1512 and represents the weeks in your study programme. Please use the weekly Learning Units to guide you in your studies.

Please consult the publication **myStudies@Unisa** which you received with your study material for more information on *myUnisa*.

4.2 Other resources – Printed support materials

Because we want you to be successful in this blended online module, we also provide you with some of the study materials in printed format. This will allow you to read the study materials, even if you are not online.

- These printed study materials will be sent to you at the beginning of the semester, but you do not have to wait to receive them to start studying – You can go online as soon as you register and all your study materials will be there.
- Therefore, the printed materials are not something that you need to wait for before you start with the module. It is only an **offline** copy of the formal content for the online module.
- This will give you the chance to do a lot of the studying for this module WITHOUT going online. This will save you money, of course, and you will be able to take as much time as you need to read -- and to re-read -- the materials and do the activities.

It is therefore very important that you log into myUnisa regularly. We recommend that you should do this at least every week, and preferably twice a week, to check for the following:

- **Check for new Announcements.** You can also set up your myLife email so that you receive the Announcement emails on your mobile device.
- **Do the Discussion forum activities.** For every unit in this module, we want you to share with the other people in your group in the activities. You can read the instructions there, and even prepare your answers but you need to go online to post your messages.
- **Do other online activities.** For some of the unit activities, you need to post something on the Blog or take a quiz or complete a survey in **Self Assessment**. Don't skip these activities because they will help you to complete the assignments and activities for the module.

We hope that this system will help you to succeed in this blended online module by giving you extra ways to study the materials and practice with all of the activities and assignments. At the same time, you **MUST** go online in order to complete the activities and assignments on time -- and to get the most from the online course.

Remember, the printed support materials are a back-up to everything that is found online, on myUnisa. There are no extra things there. **In other words, you should NOT wait for the Printed support materials to arrive to start studying.**

4.3 Prescribed books

The prescribed book for this module is:

Walter Savitch. Problem Solving with C++, 9th edition. Pearson International Edition: Addison-Wesley, 2015.

You may also use the 7th or 8th edition of the prescribed book.

You are expected to purchase your own copy of the prescribed book. For contact details of official booksellers, please consult the list of official booksellers and their addresses in

myStudies@Unisa. If you have any difficulties with obtaining books from these bookshops, please contact *vospresc@unisa.ac.za*.

We will refer to the prescribed book as Savitch.

4.4 Recommended books

You do not have to consult any other textbooks apart from Savitch. However, some of you may want to read more widely, and consult alternative references. The following useful books are available in the Unisa library. Please note that the library does not have multiple copies of these books and that only limited waiting lists are kept.

DS Malik. C++ Programming: From Problem Analysis To Program Design. Course Technology, Thomson Learning, 2009.

HM Deitel and PJ Deitel. C++ How to Program. 8 th edition. Prentice Hall, 2008.

John R. Hubbard. Programming with C++. 2 nd edition. Schaum's Outlines, 2000.

4.5 Prescribed software

The prescribed software for this module is Code::Blocks 16.01. We will refer to the software as Code::Blocks. Code::Blocks includes the MinGW C++ compiler and an Integrated Development Environment (IDE), which we use to create program files. The prescribed software is provided on the CD Disk2017 that you should have received in your study package when you registered. The Disk2017 contains instructions on how to install the software, and how to use the IDE to write, compile and execute your programs. Click on the link for COS1512 and follow the instructions.

If you did not receive Disk2017 upon registration, you should download the software immediately from myUnisa so that you are able to start with your studies at once. The software is available under Additional Resources on the COS1512 webpage.

4.6 myUnisa Forum

Content-related queries should be posted on the COS1512 discussion forum on myUnisa rather than sent to the COS1512 e-mail address. In this way fellow students can also contribute and benefit. You can also contact the e-tutor to whom you are allocated with content-related queries (see section 5.1).

4.7 Tutorial letters

In addition to the Study Guide, the software and this tutorial letter, there will be other tutorial letters during the course of the semester. Some will provide additional information (Tutorial Letters 102, 103, etc.) whilst others will discuss Assignments 1 and 2 (Tutorial Letters 201 and 202). All tutorial letters will be available on *myUnisa*. Note also that the solutions to the self-assessment assignment - Assignment 3 - are available online on the COS1511 website.

You only receive a printed copy of this tutorial letter (tutorial letter 101). All subsequent tutorial letters are only available on myUnisa under *Official Study Material*.

4.8 Additional Resources

Please check *Additional Resources* on myUnisa regularly for documents that will assist you if you have problems installing software, using the compiler, writing a program, etc. You will also find extra help for the examination there.

5 STUDENT SUPPORT SERVICES FOR THE MODULE

The Student Services Bureau of Unisa provides support for students in general academic matters, such as selecting appropriate modules, developing study skills, adapting to distance education, assistance for students with special needs or general difficulties with studies. See COSALLF/301/4/2017 and *myStudies@Unisa* for contact information. Your *myStudies@Unisa* brochure also contains other important information.

5.1 E-Tutors

Unisa offers online tutorials (e-tutoring) to students registered for modules at NQF level 5 and 6, this means qualifying first year and second year modules.

Once you have been registered for a qualifying module, you will be allocated to a group of students with whom you will be interacting during the tuition period as well as an e-tutor who will be your tutorial facilitator. Thereafter you will receive an sms informing you about your group, the name of your e-tutor and instructions on how to log onto MyUnisa in order to receive further information on the e-tutoring process. If you login into myUnisa you will notice that your group site has been added there. Your tutor will be able to assist you there. For example, if you were allocated in group 4, the group site will be named COS1512-17-S1-4E. You can use the discussion forum to discuss module content issues with your tutor as well as with students belonging to that group. You will also find the contact details of your tutor. If you have content related problems (that is, problems with the material in your study guide that you do not understand), please contact your e-tutor.

Online tutorials are conducted by qualified E-Tutors who are appointed by Unisa and are offered free of charge. All you need to be able to participate in e-tutoring is a computer with internet connection. If you live close to a Unisa Regional Centre or a Telecentre contracted with Unisa, please feel free to visit any of these to access the internet. E-tutoring takes place on myUnisa where you are expected to connect with other students in your allocated group. It is the role of the e-tutor to guide you through your study material during this interaction process. For you to get the most out of online tutoring, you need to participate in the online discussions that the e-tutor will be facilitating. Please contact your e-tutor with all content-related queries.

There are modules which students have been found to repeatedly fail, these modules are allocated face-to-face tutors and tutorials for these modules take place at the Unisa regional centres. These tutorials are also offered free of charge, however, it is important for you to register at your nearest Unisa Regional Centre to secure attendance of these classes.

This module is furthermore part of the “Science Foundation Programme” – see more information in Additional Resources.

5.2 Free computer and internet access

Unisa has entered into partnerships with establishments (referred to as Telecentres) in various locations across South Africa to enable you (as a Unisa student) free access to computers and the Internet. This access enables you to conduct the following academic related activities: registration; online submission of assignments; engaging in e-tutoring activities and signature courses; etc. Please note that any other activity outside of these are for your own costing e.g. printing, photocopying, etc. For more information on the Telecentre nearest to you, please visit www.unisa.ac.za/telecentres.

5.3 Downloading study material and software

One of the requirements for study at the School of Computing is to have regular internet access to access *myUnisa* and your *myLife* e-mails. You are therefore expected to download any study material from the Internet that, for whatever reason, is not available on paper in time. You may download it from myUnisa. The study material is updated regularly, thus you need to check the COS1512 website at least once a week on myUnisa.

Because COS1512 is a semester module, time is of the utmost importance. You should start studying the module immediately after registration. This tutorial letter, the Study Guide and the software are most important.

The **Study Guide** is available on *myUnisa* under *Additional Resources*. Please download it from *myUnisa*. It is also incorporated in the Learning Units, which you should use to guide your studies week by week.

The **software** should also be downloaded from *myUnisa* under *Additional Resources* for COS1512, at once if you do not receive Disk2017 *immediately* after registration. Please note that it is not necessary to download the full contents of the CD. You need Code::Blocks only. You may copy it onto a memory stick and install it from there according to the instructions given on the COS1512 website.

When you want to use *myUnisa* for the first time, you have to register. Go to my.unisa.ac.za and click on "Join myUnisa". Then follow the instructions on the screen. You will get a password for future use. We also suggest that you get your myLife email address as soon as possible. See the *myStudies@Unisa* brochure for instructions. The University communicates with you via this email address. You also get notified about important announcements for COS1512 via this email address. Please check your myLife email regularly.

5.4 Additional Resources on myUnisa

Apart from the Disk2017 content that is available on myUnisa under *Additional Resources*, you will find other resources such as old exam papers, extra examples of some programming constructs, etc.

5.5 Announcements on myUnisa

We urge you to access *myUnisa* on a regular basis. We put announcements on *myUnisa* regarding the module on a regular basis.

5.6 Installation of the software

Once you have access to a computer, you should install the software for this module on the computer. (If you will be using one of Unisa's computer laboratories, the software will already be installed). The software that you need for COS1512, namely a compiler and an IDE, are included on the CD-ROM disk that you should have received as part of your study package (Disk2017). Tutorial Letter MO001 contains full instructions on how to install the compiler and IDE and how to start using them.

6 MODULE-SPECIFIC STUDY PLAN

Use your myStudies@Unisa brochure for general time management and planning skills.

6.1 Syllabus

In this module we cover the following chapters of Savitch:

Chapter	Sections covered
Chapter 1	1.1 and 1.2
Chapter 4	Only 4.6
Chapter 5	Only 5.5
Chapter 6	All sections
Chapter 8	8.1 and 8.3, plus the subsection <i>Converting Between string Objects and C Strings</i> , thus excluding 8.2 with the exception of the subsection Converting Between string Objects and C Strings
Chapter 9	All sections excluding the optional subsections in 9.2
Chapter 10	All sections
Chapter 11	All sections, plus Appendixes 7 and 8
Chapter 12	12.1 and only the first two pages of 12.2
Chapter 14	14.1 and 14.2, thus excluding 14.3
Chapter 15	Only 15.1, thus excluding 15.2 and 15.3
Chapter 17	All sections

Note that some of the sections (in Chapters 1, 4 and 5) are omitted, because they have already been covered in COS1511. The other sections that are omitted fall outside the scope of this module.

6.2 Planning your academic year

In overview, the undergraduate academic year is as follows:

First semester		Second semester	
23 January	Academic year begins	10 July	Academic year begins
20 February	First assignment due	10 August	First assignment due
6 April	Second assignment due	20 September	Second assignment due
May/June	Examinations	November	Examinations

To get going with your studies, do the following:

- Read this tutorial letter (COS1512/101/3/2017) and Tutorial Letter COSALLF/301/4/2017.
- Obtain a copy of the prescribed book.

- Arrange for access to a computer.

We provide two study programmes, one for students who registered for the first semester, and one for students who registered for the second semester. We recommend that you use the study programmes as a starting point. You will probably need to adapt this schedule, taking into account your other modules and your personal circumstances.

Study programme for first semester registration:

Week	Date (Monday)	Activity	Remark
1	23 Jan	Install software, Study sections in chapters 1, 4 and 5	Do Questions 1 & 2 in Assignment 1
2	30 Jan	Study chapter 6	Do Questions 3 & 4 in Assignment 1
3	6 Feb	Study chapter 9	Do Question 5 in Assignment 1
4	13 Feb	Complete assignment 1	Due date 23 February
5	20 Feb	Study chapter 10	Do Questions 1, 2 & 3 in Assignment 2
6	27 Feb		
7	6 Mar	Study chapter 11	Do Question 4 in Assignment 2
8	13 Mar	Study chapter 12	Do Questions 5 and 6 in Assignment 2
9	20 Mar	Study chapter 15	Do Question 7 in Assignment 2 Do Questions 1 & 2 in Assignment 3
10	27 Mar	Complete assignment 2	Due date 7 April
11	3 Apr	Study chapter 17	Do Questions 3, 4 & 5 in Assignment 3
12	10 Apr	Study chapter 8	Do Questions 6 & 7 in Assignment 3
13	17 Apr	Study chapter 14	Do Questions 8 in Assignment 3
14	24 Apr	Complete assignment 3	Self-assessment
15	1 May till exam	Revision	Study <i>all</i> tutorial matter, including solutions to assignments and material provided in Additional Resources. Do examination paper supplied in examination tutorial letter <i>on paper</i> .

Study programme for second semester registration:

Week	Date (Monday)	Activity	Remark
1	11 July	Install software, Study sections in chapters 1, 4 and 5	Queries on software installation will only be answered up to 5 February Do Questions 1 & 2 in Assignment 1
2	18 July	Study chapter 6	Do Questions 3 & 4 in Assignment 1
3	25 July	Study chapter 9	Do Question 5 in Assignment 1

4	1 Aug	Complete assignment 1	Due date 11 August
5	8 Aug	Study chapter 10	Do Questions 1, 2 & 3 in Assignment 2
6	15 Aug		
7	22 Aug	Study chapter 11	Do Question 4 in Assignment 2
8	29 Aug	Study chapter 12	Do Questions 5 and 6 in Assignment 2
9	5 Sep	Study chapter 15	Do Question 7 in Assignment 2 Do Questions 1 & 2 in Assignment 3
10	12 Sep	Complete assignment 2	Due date 21 September
11	19 Sep	Study chapter 17	Do Questions 3, 4 & 5 in Assignment 3
12	26 Sep	Study chapter 8	Do Questions 7 & 7 in Assignment 3
13	3 Oct	Study chapter 14	Do Questions 8 in Assignment 3
14	10 Oct	Complete assignment 3	Self-assessment
15	17 Oct till exam	Revision	Study <i>all</i> tutorial matter, including solutions to assignments and material provided in Additional Resources. Do examination paper supplied in examination tutorial letter <i>on paper</i> .

6.3 Hints on studying this module

Study each chapter in the prescribed book by following these steps:

- Read the corresponding discussion given in the Learning Unit for the week when you have to study the chapter. This discussion is also available in the Study Guide that can be downloaded from the COS1512 website on myUnisa under Additional resources.
- Scan the chapter in Savitch to get an overview of what the chapter is about.
- Read the chapter again, making sure that you process the information. Relate the text to the given program listings. You will sometimes have to read a little ahead or read a whole section to make meaningful sense of a program listing or discussion. Many students merely read the code and not the accompanying text that explains the code.
- Remember to highlight or indicate all the words or phrases you think are key points the writer is making. You can use these and the headings to make your concept maps or summaries whichever you prefer.
- Take the source listing of the sample programs in the textbook, type it into a text file, compile it and execute it. Observe the output produced. Some of the source listings of the examples can be found on the companion website that goes with your textbook. Appendix B contains instructions on how to gain access to the source listings. The Learning Units also contain links to some of the source listings.

- Do as many as possible of the self-check questions on a section as you study it. Answers to the self-check questions are available at the end of each chapter.
- Answer the assignment questions on the chapter. Implement all programming questions on your computer.

It is important to realise that the process of learning how to program follows a learning curve: The more programs you write, the more proficient you will become. Remember that COS1512 has a large practical component and that it is essential to gain a lot of programming experience. Programming modules also require much more time than other modules with no practical work. You will probably find that you need to work hard and consistently throughout the semester to develop the necessary programming skills. Plan to **spend at least 8 hours per week on this module**.

7 MODULE PRACTICAL WORK AND WORK-INTEGRATED LEARNING

This module does not require any work integrated learning. However, all the assignments require extensive practical work on a computer. The examination is a purely written examination and does not involve doing any work on the computer. There are no compulsory separate practicals that students need to attend during the year.

All COS1512 students must have access to a computer running Windows. The computer must have a CD-ROM drive. **Note that Windows XP is the default operating system supported by Unisa.**

If you do not have a computer at home, gain access to one somewhere else, possibly at work, at a friend's home, or at one of Unisa's computer laboratories. The Unisa computer laboratories in Pretoria and at the regional offices are available to students for the practical work. You will receive a COSALL tutorial letter explaining where the laboratories are, the hours during which they are open and the booking procedure.

8 ASSESSMENT

8.1 Assessment plan

Assignments are seen as part of the learning material for this module. As you do the assignment, study the reading texts, consult other resources, discuss the work with fellow students or tutors or do research, you are actively engaged in learning. Looking at the assessment criteria given for each assignment will help you to understand what is required of you more clearly. The assessment criteria for each assignment correspond to a large extent to the learning outcomes specified in the Study Guide and Learning Units for the study material covered by the assignment.

Two sets of assignments for this year are given at the end of this tutorial letter. The first set of assignments have to be submitted by students registered for the first semester, and the second set of assignments have to be submitted by students registered for the second semester. The tutorial matter you have to master in order to complete each assignment appears in the study programme in Section 6 and at the start of each assignment. The Study Guide and Learning Units contains details on each section.

Give yourself enough time to do the assignments properly, bearing in mind that a single session in front of the computer will not be sufficient to complete a programming task. We suggest that you do the assignment question(s) on a specific chapter as soon as you have studied it. This will allow you to master the study material and to start timeously with your assignments.

The time constraints under the semester system do not allow us to accept late assignments.

All the assignments require practical work, i.e. programs that you have to implement on your computer. Submit the source code of each program, as well as the input and corresponding output for the program. Assignments 1 and 2 have to be submitted by the due date. Assignment 3 is for self-assessment, i.e. you do not have to submit it to Unisa, but will 'mark' it yourself by comparing your attempt with the model solution.

You are required to submit your assignments electronically via myUnisa in **PDF** format. Please submit only **one** PDF file for an assignment. This PDF file should contain the **source code as well as the input and the output produced by that source code** for each question in the assignment. You will find a tutorial letter that shows you how to create your assignment as a PDF file so that you can submit it electronically as well as a video on how to create a PDF file for an assignment under Additional Resources on the COS1512 website on myUnisa.

Please note the following:

- Submit only one copy of a specific assignment.
- Each assignment has a unique number. Use the correct assignment number and unique number on myUnisa when submitting your assignments electronically.
- Please make sure that the assignment you submit contains the correct content.
- **Please do not encrypt your assignment file or mark it as 'Read only'.**
- Do not send assignments directly to any of the lecturers or to the COS1512 e-mail address.
- Assignments must reach UNISA on or before the due date.
- Check on myUnisa, or contact the Assignments Section to ensure that your assignment was received by UNISA.
- All programs must be implemented on a computer. Hand-written programs will not be marked.
- Copy your programs and output to **one** WORD document and convert it to PDF before you submit.
- Use **single spacing** for the document that you submit.
- **Only PDF files will be accepted.**
- Assignments may not be submitted by fax or e-mail

For detailed information on how to submit assignments electronically, refer to the myStudies@Unisa brochure, which you received with your study package. Instructions on how to register to become a myUnisa user, are provided on the web site.

You will receive tutorial letters (201 and 202) discussing each assignment. The solution to assignment 3 is provided on the course website under Additional Resources. Work through the solutions and make sure that you understand them. When you receive your marked assignment back from Unisa, compare it to our solutions and make sure you understand the differences, and also why you lost marks. The assignments serve a very important learning function. Therefore, even if you do not submit a particular assignment, you should still complete it and compare your solution to ours as part of your study programme.

We may mark only selected questions in the assignment and not the entire assignment. However, as mentioned before, we discuss each assignment question in a detailed tutorial letter that you will receive after the due date.

When we mark assignments, we comment on your answers. Many students make the same mistakes and consequently we discuss general problems in the solutions to the assignments.

As mentioned before, it is therefore, **important to work through these tutorial letters and to make sure you understand our solutions and where you went wrong.**

The marks you obtain for an assignment are converted to a percentage. If you for instance obtained 25 marks out of a possible 50 marks for Assignment 1, you received 50% for Assignment 1. For Assignment 1 this percentage in turn contributes a weight of 20% to the year mark, and for Assignment 2 this percentage contributes a weight of 80% to the year mark.

You are welcome to work in small groups. However, every member of the group must write and submit his or her own individual assignment. Therefore, discuss the problem, find solutions, etc. in the group, but then do your own programming and submit your own effort. You will learn to program only if you sit down in front of the computer, type in the code, debug the program and get it to work. It is unacceptable for students to submit identical assignments on the basis that they worked together. That is copying (a form of plagiarism) and none of these assignments will be marked. It is dishonest to submit the work of someone else as your own, i.e. to commit plagiarism. Such unethical behaviour does not become our profession.

Assignment assessment and semester mark calculation: Your mark for this module is made up of a semester mark (20%) and an examination mark (80%). The final semester mark is calculated based on your performance in assignments throughout the semester. Therefore, assignments not only give you the opportunity to evaluate your understanding of the materials covered in the module, but also contribute towards your final mark.

The weights allocated to the assignments for COS1512 are summarized as follows:

Assignment number	Weight
1 (compulsory)	20%
2	80%
3	0% (self-assessment)

To explain how this will work, assume that a student receives 75% for assignment 1, and 80% for assignment 2. His/her year mark will then be calculated as follows:

Assignment	Mark received (Percentage)	Weight	Contribution to semester mark	
			Mark(%) * Weight(%)	Contribution
1	75%	20%	$75/100 * 20/100$	0.15
2	80%	80%	$80/100 * 80/100$	0.64
			Total:	0.79

When the total of 0.79 is converted to 20% of the final mark, it will be 15.8%, thus the student's semester mark will be 15.8%. The examination will form the remaining 80% of the final mark for the module. Note that the semester mark will not form part of the final mark for the supplementary examination.

The following formula will be used to calculate your final semester mark:

Semester mark (out of 100) x 20% + Examination mark (out of 100) x 80%

8.2 General assignment numbers

Assignments are numbered consecutively starting from 01 using Arabic numerals. The assignments are marked and a percentage is awarded according to your achievement. These assignments have a very important learning function. Please attempt (not necessarily submit) all assignments, and compare them to the solutions provided.

8.2.1 Unique assignment numbers

Assignment	Unique number
Assignment 1 Semester 1	760920
Assignment 2 Semester 1	673957
Assignment 1 Semester 2	835284
Assignment 2 Semester 2	805117

8.2.2 Due dates for assignments

The table below gives the due dates of the assignments for this module. Do not submit assignment 3 - it is a self assessment assignment.

Assignment	Due Date Semester 1	Due Date Semester 2	Weight
01	20 February	10 August	20%
02	6 April	20 September	80%
03	27 Apr - do not submit	12 October - do not submit	Self Assessment

Due to regulatory requirements (imposed by the Department of National Education) the following applies: To gain admission to the examination you have to submit Assignment 1 in time. The due date is 20 February if you are registered for the first semester and 10 August if you are registered for the second semester.

8.3 Submission of assignments

Students may submit assignments **either** by post **or** Mobile MCQ submission on your cell phone **or** electronically via myUnisa. **Assignments are not accepted via fax or email.**

For detailed information and requirements as far as assignments are concerned, see myStudies@Unisa, which you received with your study package. Follow the instructions given in Tutorial Letter COSALLF/301/4/2017, as well as the brochure myStudies@Unisa, when submitting your assignments. The URL for myUnisa is: <http://my.unisa.ac.za/>. Instructions on how to register to become a myUnisa user, and how you should format your assignments before you submit them electronically, are given on the web site. The two most important things to remember are that your submission must consist of a single PDF file, and that you may submit an assignment only once. Also, for COS512 use single line spacing in the documents that you submit.

The process to submit an assignment via myUnisa is briefly described below:

- Go to myUnisa at <https://my.unisa.ac.za/>.
- Log on with your student number and password.
- Choose the correct module (COS1512) in the orange block.
- Click on assignments in the menu on the left-hand.
- Click on the assignment number for the assignment that you want to submit.
- Follow the instructions.

PLEASE NOTE: Assignments can be tracked (e.g. whether or not the University has received your assignment or the date on which an assignment was returned to you) on myUnisa.

8.4 Assignments

FIRST SEMESTER ASSIGNMENTS ASSIGNMENT 1 (FIRST SEMESTER)

UNIQUE NUMBER	760920
DUE DATE:	20 February 2017
TUTORIAL MATTER:	Chapters 4 to 7 and 9 of the Study Guide Chapters 4 (section 4.6), 5 (section 5.5), 6 and 9 (excluding the optional parts of section 9.2) of Savitch
WEIGHT:	20%
EXTENSION:	None

Answer all the questions. Submit all the programs you are required to write, as well as the **input and output** of all programs.

Copy the programs and the required input and output to ONE word processor file with single line spacing and convert it to a PDF file before you submit it. See Additional Resources on MyUnisa for instructions on how to create a PDF file.

WE DO NOT ACCEPT ANY MEMORY STICKS OR CDs.

Question 1

Write a program that will calculate the circumference of a figure. The program must use two overloaded functions, each named `calcCircumference`. The first function must calculate the circumference of a circle, and will have one parameter of type `double` that represents the radius of the circle. The second function will have two parameters of type `double` representing the length and width of a rectangle. Both functions must calculate the circumference and return the value as type `double`.

The main function should request the user to specify whether the circumference of a rectangle or a circle must be calculated, and depending on the answer, the user must either be prompted to enter the radius of the circle, or the length and the width of the rectangle. The main function should then call the correct overloaded function and display the circumference. Define a `double` constant variable `PI` with the value of 3.14285 to use for the calculation of the circumference of the circle.

Question 2

Write a program that converts a date (dd mm) to julian format. The julian format is the day of the year the 'dd mm' date represents, e.g. the julian format of '27 03' (i.e. the 27th of March) is 31 +

28 + 27 , i.e. 31 days in January plus 28 days in February plus 27, which gives a julian date of 86. Use the following definition for the number of days in each month:

```
int daysPerMonth[12] = {31,28,31,30,31,30,31,31,30,31,30,31};
```

You can ignore leap years. The program must ask for 2 dates – read the day and month of each date separately. It must then convert both dates to their julian format, display them, and then display the number of days difference between the two dates. The program must check which is the smaller date before subtracting – the user may enter the dates in any order. Verify that the dates entered are legitimate dates, using the `assert` function and the above defined array, e.g. 31 6 is not a valid date.

Question 3

Write a program that takes its input from a file of numbers of type `double`. The program outputs to the screen the average and the standard deviation of the numbers in the file.

The file contains nothing but numbers of type `double` separated by blanks and/or line breaks. The standard deviation of a list of numbers n_1, n_2, n_3 , and so forth is defined as the square root of the average of the following numbers:

$$(n_1 - a)^2, (n_2 - a)^2, (n_3 - a)^2, \text{ and so forth}$$

The number a is the average of the numbers n_1, n_2, n_3 , and so forth.

Hint: Write your program so that it first reads the entire file and computes the average of all the numbers, and then closes the file, then reopens the file and computes the standard deviation.

Question 4

Write a program that will create userids for email addresses of a company. Names of employees will be contained in a file. Your task is to read the file character by character. Employee names are separated by semicolons. The process of creating a userid is as follows:

The first 8 characters, excluding spaces and non-alphabetical characters becomes the userid. Your program will therefore ignore all spaces and non-alphabetical characters from the input file. If the employee name excluding spaces and non-alphabetical characters consists of less than 8 characters, the userid will also contain less than 8 characters. (e.g. JG Smit will give a userid of `jgsmit` and GM Bezuidenhout will give a userid of `gmbezuid`).

You will however not work with the full surname. Your program will read character by character, and form the userid as you go along, adding each new character of the id to the output file. Call the output file `userid.dat`. As soon as you read a semicolon, the previous userid is complete and you can then write the semicolon to the output file to separate the userids. You therefore have to plan the logic properly, so that you don't have to go back to the output file to delete characters that should not be there.

Create an input file called `employee.dat` with the following data:

```
SS van der Merwe;PJ Ferreira;HW du Plessis;DF Kodisang;AA
Papoudopolous;G Mhlanga;TRF Schoeman;LJ Marais-Le Roux;CG Roux;B
Nicholaidis;TT Tshabalala;RV Mississippi;
```

Allow the user to specify the name of the input file.

Question 5

- (a) What is a pointer?
- (b) What is a dereferencing operator?
- (c) What is the difference between assignment statements `p1 = p2;` and `*p1 = *p2;`
- (d) What is a dynamic variable?
- (e) What is the purpose of the `new` operator?
- (f) What is the purpose of the `delete` operator?
- (g) What is the freestore (also called the heap)?
- (h) What is the difference between dynamic variables and automatic variables?
- (i) What is a dynamic array?
- (j) What is the advantage of using dynamic arrays?
- (k) What is the relationship between pointers and arrays?
- (l) For each of the following, write a single C++ statement that performs the identified task. Assume that variables `total` and `score` have been defined as type `int` and that `total` has been initialised to 84 and `score` to 12.
 - (i) Declare two variables `fPtr1` and `fPtr2` to be pointers to objects of type `int`.
 - (ii) Let the pointer `fPtr1` point to the `int` object `total`.
 - (iii) Print the address in memory of the `int` object `total`.
 - (iv) Add the value of `score` to the value of the object pointed to by `fPtr1`.
 - (v) Display the value of the `int` object pointed to by `fPtr1`.
 - (vi) Dynamically allocate a variable of type `int` and store its address in `fPtr2`.
 - (vii) Read a value for the variable that `fPtr2` is pointing to from the keyboard.
 - (viii) Let pointer `fPtr1` point to the same location as `fPtr2`.
 - (ix) Free the memory allocated to the variable that `fPtr1` is pointing to. What is the value of the variable that `fPtr2` is pointing to now?
 - (x) Define a pointer type `IntPtr` for pointer variables that contain pointers to `int` variables.
 - (xi) Define a pointer variable `p` of type `IntPtr` that can point to an `int` variable.
 - (xii) Dynamically allocate an array of 10 elements of type `int` and store its address in `p`.

- (xiii) Initialise all the elements of the array that p is pointing to, to 0 .
- (xiv) Free the memory allocated to the array that p is pointing to.
- (m) Write a program that asks a user to enter the size of a dynamic array that stores scores obtained by students. Create the dynamic array and a loop that allows the user to enter a score into each array element. Loop through the array, find the maximum score and output it. Delete the memory allocated to your dynamic array before exiting your program.

ASSIGNMENT 2 (FIRST SEMESTER)

UNIQUE NUMBER	673957
DUE DATE:	6 April 2017
TUTORIAL MATTER:	Chapters 10, 11, 12 and 15 of the Study Guide (Appendix D) Chapters 10, 11, 12 (excluding “Creating a Namespace”) and 15 (only 15.1 “Inheritance basics”) Appendices 7 and 8 in Savitch
EXTENTION:	None
WEIGHT:	80%

Answer all the questions. Submit all the programs you are required to write, as well as the **input and output** of all programs.

Copy the programs and the required input and output to ONE word processor file with single line spacing and convert it to a PDF file before you submit it. See Additional Resources on MyUnisa for instructions on how to create a PDF file.

WE DO NOT ACCEPT ANY MEMORY STICKS OR CDs.

Question 1

Consider the following structure used to keep record of a student's scores:

```
struct Student
{
    string name;
    int quiz1;
    int quiz2;
    int midtermExam;
    int finalExam;
}
```

A student is graded according to the following policies:

1. The two quizzes are each marked out of a maximum of 10.
2. The midterm exam and the final exam are each marked out of a maximum of 100.
3. The final exam counts for 50% of the grade, the midterm counts for 25%, and the two quizzes together count for a total of 25%. (Do not forget to normalize the quiz scores, i.e. they should be converted to a percentage before they are averaged in.)

Turn the student record into a class type rather than a structure type. The student record class should have member variables for all the input data. All member variables should be `private`. Include `public` member functions for each of the following:

- a default constructor that sets the student 's name to a blank string, and all the scores to 0;
- member functions to set each of the member variables to a value given as an argument to the function (i.e. mutators);
- member functions to retrieve the data from each of the member variables (i.e. accessors);
- and a function that calculates and returns the student's weighted average numeric score for the entire course.

Use this class in program which grades a student. The program should read in the student's name and scores and output the student's record as well as the student's average numeric score for the entire course. Use the keyboard to supply input and display the output on the screen. Test your program with the following input:

Student name: Johnny Deppo
 Quiz 1: 7
 Quiz 2: 5
 Midterm exam: 65
 Final exam: 73

Question 2 – a bit of theory and terminology

- What is the purpose of the keywords `public` and `private` in the class declaration?
- What is the difference between a class and an object?
- What does it mean to 'instantiate' an object?
- What is the purpose of a constructor?
- What is the difference between the default constructor and the overloaded constructor?
- What is the purpose of a destructor?
- What is the purpose of an accessor?
- What is the purpose of a mutator?
- What is the purpose of the scope resolution operator?
- What is the difference between the scope resolution operator and the dot operator?
- What is the difference between a member function and an ordinary function?
- What is an abstract data type (ADT)?
- How do we create an ADT?
- What are the advantages of using ADTs?
- What is separate compilation?
- What are the advantages of separate compilation?
- What is a derived class?
- What is the purpose of inheritance?

Question 3

3 (a) Consider the following class declaration and write code to correct it:

```
class Employee
{
public:
    Employee ();
    string getName();
private:
    bool citizen;
    string lastName;
    string firstName;
    string employeeNumber;
};
```

Explain what is wrong with the following code fragment and write code to show how to correct it:

```
int main()
{
    Employee e;
    .....
    string eNumber = e.employeeNumber;
    return 0;
}
```

3 (b) Consider the following code fragment. Explain what is wrong with it and write code to correct it:

```
Employee myCompany[5], director;
director.citizen = myCompany.citizen[0];
director.lastName = myCompany.lastName[0];
director.firstName = myCompany.firstName[0];
director.employeeNumber = myCompany.employeeNumber[0];
```

Question 4

The GreatClassics DVD store has a very large collection of old movies on DVD for rental use. Design and implement a C++ class called `DVD` that handles information regarding the DVDs in the GreatClassics DVD store. Think of all the things you would want to do with such a class and write corresponding member functions for your `DVD` class. Your class declaration should be well-documented so that users will know how to use it.

Write a main program that does the following:

- Declare an array to hold information for the DVDs available in the GreatClassics DVD store. The elements of the array must be of type `DVD`.
- You must have member variables to hold the title of a DVD, the name of the director, the year released, number of copies in stock, the running time in minutes, etc. Initialise the array with applicable information.

- The owner of the GreatClassics DVD store decides that he will offer a discount of 15% on all DVDs produced before 1960, unless they have a running time of longer than two hours. Display the titles, directors, year of release and running time of all DVDs produced before 1960. Now determine and display the number of DVDs that will be discounted and list their titles.

Enrichment exercises:

- Turn your DVD class into an ADT, so that separate files are used for the interface and implementation. Use separate compilation to compile the implementation separate from the application file that tests the ADT.
- Adapt the application program to use a vector instead of an array. It should not be necessary to change the class interface or implementation file in any way.

PLEASE NOTE: The enrichment exercises do not form part of the assignment. It is for practice only.

Question 5

Define a class `Player` as an ADT that uses separate files for the interface and the implementation. The class represents a Player in a video game. This class has the following data members:

```
string name; // the name of the player
string team; // the team for which this player plays
int level; // the level to which the player has advanced
// to in the game.
int points; // the number of points accumulated
```

The class should contain a default constructor that initializes `name` to "*Player 0*", `team` to "*Team 0*", `level` to 0 and `points` to 0. It should also contain an overloaded constructor that accepts four parameters to set the `name`, `team`, `level` and `points` to specified values. The destructor should output "*Game Over*".

Include accessor functions that return the values stored in each of the member variables of an object of class `Player` (i.e. `get` functions), as well as mutator functions to update each of the member variables of an object of class `Player` respectively (i.e. `set` functions with a parameter to set each member variable to a value specified by the parameter). The class should also contain a `void` member function called `reset()` that resets the member variables of a `Player` to values specified by parameters.

Overload the equality operator `==` as a friend function for class `Player`. This function returns `true` if the `team` member variable of `Player1` is identical to that of `Player2` and `false` otherwise. Use the following prototype:

```
bool operator==(const Player & Player1, const Player & Player2)
```

Overload the comparison operator `>` as a friend function for class `Player`. This function returns `true` if the `points` member variable of `Player1` is bigger than that of `Player2` and `false` otherwise. Use the following prototype:

```
bool operator>(const Player & Player1, const Player & Player2)
```

Define an overloaded prefix operator `++` (implemented as a `friend` function) to return the current instance of the class `Player` after incrementing the `points` by 1. For every 100 `points` scored, the `level` is also incremented by 1. Hint: Use the `%` (modulus) operator.

A player can request hints while playing the game. For every hint requested, 10 `points` should be deducted from a player's `points`, and the `level` (if necessary) at which (s)he plays adapted accordingly. To implement this, define a void member function `requestHint()` to print a message *"Please give me a hint"* (We will not try to supply the hints themselves at this stage or worry about how this will be done.) Member function `requestHint()` should then adapt the `points` and `level` by calling the overloaded prefix operator `--` to return the current instance of the class `Player` after decrementing the `points` by 10 and adapting the `level` accordingly. You should also implement the overloaded prefix operator `--` as a `friend` function.

Test your class by writing a program to do the following:

- Overload the stream extraction operator `>>` and the stream insertion operator `<<` as `friend` functions for class `Player`. The stream insertion operator `<<` should display the name, team, level and points of a player.
- Use the default constructor to instantiate two objects `player1` and `player3`.
- Use the overloaded constructor to instantiate an object `player2` with the following values for the member variables:
 - Name: "Jane"
 - Team: "BlueSwallows"
 - Level: 1
 - Points: 99
- Use the overloaded stream insertion operator `<<` to display the details of `player1` and `player2`.
- Reset `player1` with the following values for the member variables:
 - Name: "Peter"
 - Team: "BlueSwallows"
 - Level: 2
 - Points: 109
- Use the overloaded stream insertion operator `<<` to display the details of the new `player1`.
- Let `player1` request a hint.
- Add 1 point to `player2`'s marks, by using the overloaded prefix operator `++`.
- Again use the overloaded stream insertion operator `<<` to display the details of `player1` and `player2`.
- Check whether the two players are in the same team or not (use the overloaded `==` operator), and display the result including both players' names and the team name if they are in the same team.
- Determine which player has won by using the overloaded `>` operator and display the name of the player that has won.
- Use the overloaded stream extraction `>>` operator to obtain the details for `player3`. Use the following information:
 - Name: Alex
 - Team: Redsoxc
 - Level: 3
 - Points: 209

- Use the overloaded stream insertion operator << to display the details of `player3`.
- Change the team for which `player3` is playing to "BlueSwallows"
- Use the overloaded stream insertion operator << to display the new details of `player3`.

Question 6

A running club keeps information on record about each of its members. Define a class `Runner` as an ADT that uses separate files for the interface and the implementation. This class represents a one runner's record. For each `Runner` the following information should be kept as member variables:

number, name, age, sex (a `char` member variable), whether or not the runner has qualified for entering the Comrades marathon (a `bool` member variable), an array with the distances (in kilometers) of each of the last five races the runner ran, and a corresponding array with the times (in seconds) the runner took to complete each of these five races.

The class should contain a default constructor that initialises all the member variables with string values to empty strings and all the member variables with numeric values to 0. The `bool` member variable should be initialised to `false`. The destructor should print a message "Bye!".

Class `Runner` has accessor and mutator functions that returns / sets the value of the member variables for the runner's number, name, age, sex and Comrades qualification. Also include a private member functions to convert time in hours, minutes and seconds to seconds only.

Class `Runner` has a member function `update()` to remove the first element in the two arrays that keeps the race distance and corresponding time, then move all subsequent elements one place up in the array and add a new element with the distance and time of the last race run in the last element of each of the two arrays.

To qualify to run the Comrades marathon, a runner needs to have completed either a half marathon (21 km) in less than 90 minutes or a marathon (42 km) in less than 3 hours (180 minutes) during the preceding six months. Include a `void` member function `runComrades()` in class `Runner` to check whether or not a runner will be allowed to enter the Comrades. Member function `runComrades()` should only check the last race that has been run and update the `bool` member variable indicating whether or not the runner has qualified for entering the Comrades marathon, accordingly. NB: Note that it may be possible that a runner has already qualified for the Comrades, but that the last race does not allow the runner to qualify. In this case the member variable indicating whether or not the runner has qualified for entering the Comrades marathon should not be changed.

Add a member function `displayInfo()` to display a runner's information. Member function `displayInfo()` display a runner's number, name, age, name of the sex, whether or not the runner has qualified for entering the Comrades marathon, the distances of the last five races run and the corresponding times (in hours, minutes and seconds).

Overload the stream extraction operator >> (implemented as a **friend** function) so that it can be used to input an object of class `Runner` from a file.

Overload the stream insertion operator << (implemented as a **friend** function) so that it can be used to output an object of class `Runner` containing values for all the member variables of class `Runner` to a file.

Demonstrate the class in an application program (`main()`) to test your class that does the following:

Extract one object of class `Runner` from a file (`Runners.dat`), display the runner's number and name on the screen, and request the user to enter the detail of the last race the runner had run, i.e. the distance and time taken to complete the race. Use member function `update()` to update the two arrays keeping the race distance and corresponding time.

Use the member function `runComrades()` to check whether or not the runner has qualified for entering the Comrades marathon and update the member variable indicating whether or not the runner has qualified for entering the Comrades marathon accordingly.

Once a runner's record has been updated, use member function `displayInfo()` to display a runner's info on the screen, and output the record to a file `RunnerUpdated.dat`.

Your program should also determine how many men and how many women have qualified to enter the Comrades.

Use the following as the contents for the file `Runners.dat`:

```
1 John Martin
23 M 1
21 42 42 21 42
4680 8700 11040 5640 10900
2 Peter Monyane
26 M 1
42 42 42 42 42
11000 11070 10960 10090 9090
3 Sarah Sekonyane
31 F 0
21 21 21 42 21
5600 5499 5555 12000 6000
4 Sean Naidoo
35 M 0
21 21 42 21 21
6000 5600 11900 5500 6120
```

The first four lines represent the record kept for the first runner, John Martin. His number is 1, he is 23 years old, male and has already qualified for the Comrades (see line 2). He has run two half marathons (21 km) and three full marathons (42 km). The fourth line contains the times he has taken to complete each race in seconds.

Assume the race results used to update the runners records is for a half marathon (21 km). John Martin completed it in 2 hours, 2 minutes and 2 seconds; Peter Monyane completed it in 1 hour 40 minutes and 21 seconds; Sarah Sekonyane completed it in 1 hour 29 minutes and 22 seconds (just qualifying!); and Sean Naidoo completed it in 1 hour 40 minutes and 23 minutes.

Question 7

Define a class named `Book` with member variables for the book's title, author, price and the number of copies in stock. Add appropriate constructors and accessors for class `Book`. Include the following member functions:

- Member function `CalcSalePrice()` to calculate a new price for a book when it is on sale, based on a discount percentage provided by the shop owner. Member function `CalcSalePrice()` should also change the book's price accordingly.
- Member function `BookSold()` to reduce the number of copies in stock by the number of copies sold, when a book is sold.
- Member function `DisplayInfo()` to display the name of the book, the author, the price and the number of copies in stock.

(a) Implement class `Book`.

(b) Test class `Book` in a driver program that does the following:

- Instantiate an object of class `Book` with the following details:
 Title: Create a beautiful life
 Author: John Apfelbaum
 Price: R395.99
 Number of copies: 12
- Use the accessor member functions to display the specifications of the instantiated object on the screen.
- The book goes on sale with a 15% discount. Calculate a new price for the book with member function `CalcSalePrice()`.
- Two copies of the book are sold. Update the number of copies in stock with member function `BookSold()`.
- Use member function `DisplayInfo()` to display the current information about the book.

(c) Derive a class `TextBook` from class `Book` with two additional member variables `subject` and `ISBN`. Class `TextBook` should override member function `DisplayInfo()` to include the subject and ISBN numbers when displaying the specifications of a `Textbook` object. Implement the overloaded constructor for class `Textbook` by invoking the base class constructor.

(d) Test class `TextBook` in a driver program that does the following:

- Instantiate an object of class `TextBook` with the following details:
 Title: Easy C++
 Author: Annie Johnson
 Price: R495.99
 Number of copies: 42
 Subject: Computer Science
 ISBN: 123456789
- Use the accessor member functions to display the specifications of the instantiated object on the screen.

- The book goes on sale with a 10% discount. Calculate a new price for the book with member function `CalcSalePrice()`.
- One copy of the book is sold. Update the number of copies in stock with member function `BookSold()`.
- Use member function `DisplayInfo()` to display the current information about the book.

SECOND SEMESTER ASSIGNMENTS
ASSIGNMENT 1 (SECOND SEMESTER)

UNIQUE NUMBER	835284
DUE DATE:	10 August 2017
TUTORIAL MATTER:	Chapters 4 to 7 and 9 of the Study Guide Chapters 4 (section 4.6), 5 (section 5.5), 6, and 9 (excluding the optional parts of section 9.2) of Savitch
WEIGHT:	20%
EXTENSION:	None

Answer all the questions. Submit all the programs you are required to write, as well as the **input and output** of all programs.

Copy the programs and the required input and output to ONE word processor file with single line spacing and convert it to a PDF file before you submit it. See Additional Resources on MyUnisa for instructions on how to create a PDF file.

WE DO NOT ACCEPT ANY MEMORY STICKS OR CDs.

Question 1

At the annual school bazaar, apart from other prizes, ticket numbers can also win prizes. Write a program to determine whether a ticket number is a winning number, and calculate the cash prize for the ticket number. The program should use two overloaded functions, each named `getPrize()`. The first function receives as parameters, the ticket number and the gender of the ticket holder (a character, 'f' or 'm'). The second function receives as parameters the ticket number and the age of the ticket holder (as an integer value).

To test the overloaded functions, in the main program, the user must be asked to input the ticket number. If the ticket number is divisible by 100, the user must be asked to key in his/her gender. Function `getPrize()` must then be called to calculate the prize money. If the gender is male, and the ticket number is greater than 30000, the prize money is the ticket number divided by 90.00. If the gender is female, and the ticket number is greater than 20000, the prize money is the ticket number divided by 80.00. Alternatively, if the ticket is not divisible by 100, but divisible by 7 and divisible by 6, the user must be asked to input his/her age. The second function `getPrize()` calculates the prize money as follows: for up to an age of 21, the prize money is the age multiplied by 40. For an age over 21 the prize money is the age multiplied by 30. Both functions should return the prize money that should be displayed on the screen. If the ticket number does not adhere to the rules, a message should be displayed to say that the ticket is not a winning ticket.

Question 2

Write a program that determines the difference between two times in 24h00 format. The program must ask the user to enter two times in 24h00 format, convert the two times to seconds, get the difference, and display the result in 24h00 format again. The 24h00 format time should be input as hh mm ss, and displayed as hh:mm:ss, e.g. 13 hours, 12 minutes and 34 seconds will be input as 13 12 34 and displayed as 13:12:34.

The program must check which time is the smaller time before subtracting – the user may enter the times in any order. Verify that the times entered are legitimate times, using the `assert` macro, e.g. 23 65 01 is not a valid time.

Question 3

Compute the median of a data file. The median is the number that has the same number of data elements greater than the number as there are less than the number. For the purposes of this problem you are to assume that the data is sorted (that is, is in increasing order). The median is the middle element of the file if there are an odd number of elements, or the average of the middle two elements if the file has an even number of elements. You will need to open the file, count the number of elements, close the file and calculate the position of the middle of the file, open the file again, count up to the entries you need, and calculate the middle.

Test your program with the following two data files:

File 1:

1 4 6 12 14 18 29 33 37 40 45 47 49 51 55 56 59 60 63

File 2:

3 6 7 9 13 16 19 21 26 33 39 41 47 51 58 65 77 80

Question 4

Write a program that reads text from a file and encrypts it by using a simple technique. The encryption is done simply by replacing each character in the file with its successor. For instance, “a” is replaced with “b” or “c” is replaced with “d”. A “z” should be replaced with an “a”. The program must be able to handle both lowercase and capital letters. It should also be able to decrypt the encrypted file by replacing each character of the encrypted file with its predecessor. For instance, “b” is replaced with “a” or “d” is replaced with “c”. An “a” should be replaced with a “z”. The program should give the user a choice between encrypting or decrypting a file. If the user chooses either of the options, the program should allow the user to specify the names of the input and output files, and then do the process according to the option chosen by the user. It should then save the result to the specified output file name. Display both the input file and the output file on the screen after the processing has been done.

Question 5

- (b) What is a pointer?
- (b) What is a dereferencing operator?
- (c) What is the difference between the assignment statement `p1=p2;` and `*p1 = *p2;`
- (d) What is a dynamic variable?

- (e) What is the purpose of the `new` operator?
- (f) What is the purpose of the `delete` operator?
- (g) What is the freestore (also called the heap)?
- (h) What is the difference between dynamic variables and automatic variables?
- (i) What is a dynamic array?
- (j) What is the advantage of using dynamic arrays?
- (k) What is the relationship between pointers and arrays?
- (l) For each of the following, write a single C++ statement that performs the identified task. Assume that variables `salary` and `increase` have been defined as type `double`, and that `salary` has been initialised as `4500.00` and `increase` as `475.00`.
 - i. Declare two variables `fPtr1` and `fPtr2` to be pointers to objects of type `double`.
 - ii. Let the pointer `fPtr2` point to the `double` object `salary`.
 - iii. Let the pointer `fPtr1` point to the `double` object `increase`.
 - iv. Print the address of the object pointed to by `fPtr1`.
 - v. Print the value of the object pointed to by `fPtr2`.
 - vi. Dynamically allocate a variable of type `double` and store its address in `fPtr2`.
 - vii. Increase the value of the object that `fPtr2` is pointing to, by the value that `fPtr1` is pointing to, only if `salary` is greater than `4200.00`.
 - viii. Let pointer `fPtr1` point to the same location as `fPtr2`.
 - ix. Free the memory allocated to the variable that `fPtr2` is pointing to. What is the value of the variable that `fPtr1` is pointing to now?
 - x. Define a pointer type `DoublePtr` for pointer variables that contain pointers to `double` variables.
 - xi. Define a pointer variable `pd` of type `DoublePtr` that can point to a `double` variable.
 - xii. Dynamically allocate an array of 20 elements of type `double` and store its address in `pd`.
 - xiii. Free the memory allocated to the array that `pd` is pointing to.
- (m) Write statements to do the following:
 - i. Declare an `int` variable `count`, ask the user to input its value, and read in the value.
 - ii. Declare `p1` to be a pointer to a variable of type `int`.
 - iii. Dynamically allocate an array of `count` integers and store its address in `p1`.

- iv. Initialise the elements of the array that `p1` is pointing to, so that each element has the value of its position in the array (starting at position 0 as the first position).
 - v. Display the values of the array elements – each element on a new line.
 - vi. Free the memory allocated to the array that `p1` is pointing to.
- (m) Write a program that asks a user to enter the size of a dynamic array that stores scores obtained by students. Create the dynamic array and a loop that allows the user to enter a score into each array element. Loop through the array to calculate the average score and output it. Delete the memory allocated to your dynamic array before exiting your program.

ASSIGNMENT 2 (SECOND SEMESTER)

UNIQUE NUMBER	805117
DUE DATE:	20 September 2017
TUTORIAL MATTER:	Chapters 10, 11, 12 and 15 of the Study Guide (Appendix D) Chapters 10, 11, 12 (excluding “Creating a Namespace”) and 15 (only 15.1 “Inheritance basics”) Appendices 7 and 8 in Savitch
EXTENTION:	None
WEIGHT:	80%

Answer all the questions. Submit all the programs you are required to write, as well as the **input and output** of all programs.

Copy the programs and the required input and output to ONE word processor file with single line spacing and convert it to a PDF file before you submit it. See Additional Resources on MyUnisa for instructions on how to create a PDF file.

WE DO NOT ACCEPT ANY MEMORY STICKS OR CDs.

Question 1

Consider the following structure used to keep employee records:

```
struct Employee
{
    string firstName;
    string lastName;
    float salary;
}
```

Turn the employee record into a class type rather than a structure type. The employee record class should have private member variables for all the data. Include public member functions for each of the following:

- a default constructor that sets the employee’s first name and last name to a blank string, and his annual salary to 0;
- an overloaded constructor that sets the member variables to specified values;
- member functions to set each of the member variables to a value given as an argument to the function (i.e. mutators);
- member functions to retrieve the data from each of the member variables (i.e accessors);

Embed your class definition in a test program. The test program should create two Employee objects and display each object’s annual salary. Use the overloaded constructor to initialise one

of the Employee records as Joe Soap with an annual salary of R1456.00. Obtain the following input values for the other Employee record from the keyboard:

Joanne Soape
R15446.66

Now give each employee a 10% raise and display each `Employee` object's annual salary again.

Question 2 – a bit of theory and terminology

- (a) What is the purpose of the keywords `public` and `private` in the class declaration?
- (b) What is the difference between a class and an object?
- (c) What does it mean to 'instantiate' an object?
- (d) What is the purpose of a constructor?
- (e) What is the difference between the default constructor and the overloaded constructor?
- (f) What is the purpose of a destructor?
- (g) What is the purpose of an accessor?
- (h) What is the purpose of a mutator?
- (i) What is the purpose of the scope resolution operator?
- (j) What is the difference between the scope resolution operator and the dot operator?
- (k) What is the difference between a member function and an ordinary function?
- (l) What is an abstract data type (ADT)?
- (m) How do we create an ADT?
- (n) What are the advantages of using ADTs?
- (o) What is separate compilation?
- (p) What are the advantages of separate compilation?
- (q) What is a derived class?
- (r) What is the purpose of inheritance?

Question 3

3 (a) Consider the following class declaration:

```
class Person
{
public:
    Person();
    string getName();
private:
```

```

    string ID;
    string lastName;
    string firstName;
    string phoneNumber;
};

```

Explain what is wrong with the following code fragment and include code to correct it:

```

int main()
{
    Person p;
    .....
    string pNumber = p.phoneNumber;
    return 0;
}

```

3 (b) Consider the following declaration. Explain what is wrong with it and write code to correct it:

```

Person me, friends[10];
for (int i = 0; i <10; i++)
{
    if (me.firstName == friends.firstname[i])
        cout << "My friend no " << i
            << " has the same first name as me!";
}

```

Question 4

Design and implement a C++ class called `Book` that handles information regarding the books in a bookshop. Think of all the things you would want to do with such a class and write corresponding member functions for your `Book` class. Your class declaration should be well-documented so that users will know how to use it.

Write a main program that does the following:

- Declare an array to hold information for the books available in the bookshop. The elements of the array must be of type `Book`.
- You must have member variables to hold the title of the book, the name of the author, the price of a book (in Rands), number of copies in stock, etc. Initialise the array with applicable information.
- The owner of the bookshop decides that he will offer a discount of 15% on all books of which he has more than 20 copies in stock. Determine the number of books of which he has more than 20 copies in stock, and display their titles, authors and current price. Then adapt the price of all books of which he has more than 20 copies in stock. Now display the titles, authors and the price of **all** books available in the store, including those that have been discounted.

Enrichment Exercise:

- (a) Turn your `Book` class into an ADT, so that separate files are used for the interface and implementation. Use separate compilation to compile the implementation separate from the application file that tests the ADT.
- (b) Adapt the application program to use a vector instead of an array. It should not be necessary to change the class interface or implementation file in any way.

PLEASE NOTE: The enrichment exercises do not form part of the assignment. It is for practice only.

Question 5

Define a class `Team` as an ADT that uses separate files for the interface and the implementation. The class represents a team in the World Cup soccer tournament. This class has the following data members:

```
string country; // the name of the country for which this team plays
int round; // the round in which the team currently plays
int points; // the points the team has accumulated
int goalsFor; // the goals the team has scored
int goalsAgainst; // the goals scored against the team
```

The class should contain a default constructor that initializes `country` to "*Country 0*"; and `round`, `points`, `goalsFor` and `goalsAgainst` to 0. It should also contain an overloaded constructor that accepts five parameters to set the `country`, `round`, `points`, `goalsFor` and `goalsAgainst` member variables to specified values.

The destructor should output "*Game Over*".

Include accessor functions that return the values stored in each of the member variables of an object of class `Team` (i.e. `get` functions), as well as mutator functions to update each of the member variables of an object of class `Team` respectively (i.e. `set` functions with a parameter to set each member variable to a value specified by the parameter).

The class should also contain a `void` member function called `reset()` that resets the member variables of a `Team` to values specified by parameters.

In addition the class also have member functions `calcGoalDifference()` and `update()`. Member function `calcGoalDifference()` calculates the difference between the number of goals scored by the team and the number of goals scored against the team. Member function `update()` updates the `points`, `goalsFor` and `goalsAgainst` member variables by adding the function's parameter values to the points for a team as well as to the goals scored by the team and against the team.

Overload the equality operator `==` as a `friend` function for class `Team`. This function returns `true` if both the `points` member variable and the goal difference of `Team1` is identical to that of `Team2` and `false` otherwise. Use the following prototype:

```
bool operator==(const Team & Team1, const Team & Team2)
```

Overload the comparison operator `>` as a `friend` function for class `Team`. This function returns `true` if the `points` member variable of `Team1` is bigger than that of `Team2`; or if the `points`

member variable of `Team1` is equal to that of `Team2` and the goal difference of `Team1` is bigger than that of `Team2`. Otherwise the function returns `false`. Use the following prototype:
`bool operator>(const Team & Team1, const Team & Team2)`
 Define an overloaded prefix operator `++` (implemented as a friend function) to return the current instance of the class `Team` after incrementing the `round` member variable by 1.

Overload the stream extraction operator `>>` and the stream insertion operator `<<` as friend functions for class `Team`. The stream insertion operator `<<` should display the country, round, points, goals for and goals against the team, for a team.

Test your class by writing a program to do the following:

- Use the default constructor to instantiate objects `opponent` and `newOpponent` of class `Team`.
- Use the overloaded constructor to instantiate an object `home` of class `Team` by initializing the `country` member variable to "South-Africa", the `round` member variable to 1, the `points` member variable to 4, the `goalsFor` member variable to 6 and the `goalsAgainst` member variable to 4.
- Reset the `opponent` object to the following values: "Germany" 1 4 6 4.
- Use the overloaded insertion operator `<<` to display the values of the member variables of objects `home` and `opponent`.
- Use the overloaded equality operator `==` to determine whether `home` and `opponent` has the same number of `points` and the same goal differences. If so, display a message "This is a tie!". If it is not a tie, use the overloaded comparison operator `>` to determine which of `home` and `opponent` has the most points; and then increment the `round` member variable of the appropriate object (`home` or `opponent`) by using the overloaded prefix operator `++`.
- Use the accessor functions to display the values of the `country`, `points` and `round` data members of both the `home` and `opponent` objects.
- Now use the overloaded stream extraction operator `>>` to obtain values for the member variables for `newOpponent`. Use the following values as input: country: "Spain"; round: 1; points: 7; goalsFor: 8; and goalsAgainst: 2.
- South-Africa has won the match against Spain with 2 goals against 0. Obtain the appropriate values from the user and update the `points`, `goalsFor` and `goalsAgainst` member variables of South-Africa (`home`) and Spain (`newOpponent`) accordingly. A win counts 3 points and a loss 0.
- Once again, use the overloaded equality operator `==` to determine whether `home` and `newOpponent` has the same number of `points` and the same goal differences. If so, display a message "This is a tie!", otherwise use the overloaded comparison operator `>` to determine which of `home` and `newOpponent` has the most points; and then increment the `round` member variable of the appropriate object (`home` or `newOpponent`) by using the overloaded prefix operator `++`.
- Use the accessor functions to display the values of the `country`, `points` and `round` data members of both the `home` and `newOpponent` objects.

Question 6

A college offers diploma course with five set modules each. Define a class `Student` as an ADT that uses separate files for the interface and the implementation. This class represents a record of one student's registration and results. For each `Student` the following information should be kept as member variables:

the student's name, ID, student number, the name of the diploma, average mark, an array with the five module codes for the diploma, and another array with five results - one for each of the modules.

Class `Student` should also have a member function `calcAverage()` to calculate the average for the five results and a member function `pass()` to determine whether or not a student has passed the module. A student only passes a diploma course if he or she has obtained more than 50% for all the modules, regardless of his or her average mark. Member function `pass()` should return a boolean value.

Add a member function `displayResults()` to display a student's results after the examinations. Member function `displayResults()` display a student's name, student number, name of the diploma registered for, his or her results for each module, the average mark and whether he or she passed or failed the diploma.

In addition, the class should contain a default constructor that initializes all the member variables with string values to empty strings and all the member variables with numeric values to 0. The destructor should print a message "Bye!".

Include accessor and mutator functions that returns / sets the value of the member variables for the student's name, ID, student number, the name of the diploma, and average mark. The mutator that sets the diploma name, also sets the module names for the diploma automatically. At the moment the college only offers two diplomas, Garden Design and Gourmet Cooking. Garden Designs students take modules G1, G2, G3, G4 and G5, while Gourmet Cooking students take modules C1, C2, C3, C4 and C5. Add an accessor function to return the name of one module and a mutator function to set the value of one exam result.

Overload the stream insertion `<<` (implemented as a **friend** function) so that it can be used to output an object of class `Student` containing values for all the member variables of class `Student` to a file.

Demonstrate the class in an application program (`main()`) to test your class that does the following:

Initialise an array of three objects of class `Student` with the following values:

Name	John Martin	Busi Molefe	Sean Naidoo
ID	78120189	81011201	69812018
Student number	12345	23456	34567
Diploma	Garden Design	Gourmet Cooking	Garden Design

Display one `Student` record at a time (name, ID, student number and course) and capture the exam results for each module for the student from the keyboard. Calculate the average mark for the student. Once a student's results have been captured, output the record to a file `RegisteredStudentsResults`. Once all the results for all the students have been captured, use the member function `displayResults()` to display the results for each student.

Use the following input data to test your class (each column represents the five results for the five modules the student in the first row of that column took):

John Martin	Busi Molefe	Sean Naidoo
55	69	69
65	71	45
59	66	66
68	59	57
60	63	50

Also include a copy of your `RegisteredStudentsResults` file with your assignment.

Question 7

Define a class named `Prescription` with member variables for the prescription itself (i.e. the medicine prescribed), the patient's name, the medical aid fund, the medical aid number and cost. Add appropriate constructors and accessors for class `Prescription`. Include the following member functions:

- Member function `Discount()` to calculate a discount for prescriptions for members of THEAID medical aid, based on a discount percentage provided by the shop owner. Member function `Discount()` should then change the cost accordingly.
- Member function `DisplayInfo()` to display the prescription itself, the patient's name, the medical aid fund, the medical aid number and cost.

(a) Implement class `Prescription`.

(b) Test class `Prescription` in a driver program that does the following:

- Instantiate an object of class `Prescription` with the following details:

Medicine prescribed: Panado

Patient: John Apfelbaum

Medical aid fund: THEAID

Medical aid number:12345

Cost: R39.59

- Use the accessor member functions to display the specifications of the instantiated object on the screen.
- Use member function `Discount()` to determine whether discount should be given, and if so, adapt the cost accordingly.
- Use member function `DisplayInfo()` to display the current information about the book.

(c) Derive a class `RepeatPrescription` from class `Prescription` with two additional member variables `numberOfRepeats` and `lastDateIssued`. Also add a member function `issuePrescription()` to update the date the prescription is issued and decrease the member variable `numberOfRepeats` by one. When member variable

`numberOfRepeats` becomes 0, a message should also be displayed to indicate that this is the last issue of the current prescription.

Class `RepeatPrescription` should override member function `DisplayInfo()` to include the number of repeats and the last date the prescription was issued when displaying the specifications of a `RepeatPrescription` object. Implement the overloaded constructor for class `Prescription` by invoking the base class constructor.

(d) Test class `RepeatPrescription` in a driver program that does the following:

- Instantiate an object of class `RepeatPrescription` with the following details:

Medicine prescribed: Myprodol

Patient: Annie Apfelbaum

Medical aid fund: MYAID

Medical aid number: 43215

Cost: R89.59

Number of repeats: 1

Last date issued: 20170620

- Use the accessor member functions to display the specifications of the instantiated object on the screen.
- Use member function `issuePrescription()` to issue and update the repeat prescription.
- Use member function `DisplayInfo()` to display the current information about the repeat prescription.

SELF ASSESSMENT ASSIGNMENT 3 (SEMESTER 1 AND SEMESTER 2)

TUTORIAL MATTER:	Chapters 8, 14, 15 and 17 of the Study Guide (Appendix D) Chapters 8, 14 (excluding section 14.3), 15 (excluding sections 15.2 and 15.3) and 17 of Savitch
WEIGHT:	None

This assignment is for **self-assessment**. Do not submit this assignment. The **solution** to this assignment appears in **Appendix C** of this tutorial letter.

Question 1

Examine the code fragment below and answer the questions that follow:

```
1: #include <iostream>
2: using namespace std;
3:
4: //-----
5:
6: class A
7: {
8:     private:
9:         int x;
10:    protected:
11:        int getX();
12:    public:
13:        void setX();
14:};
15:
16:int A::getX()
17:{
18:    return x;
19:}
20:
21:void A::setX()
22:{
23:    x=10;
24:}
25:
26://-----
27:class B
28:{
29:    private:
30:        int y;
```

```

31:  protected:
32:      A objA;
33:      int getY();
34:  public:
35:      void setY();
37:};
38:
39: void B::setY()
40: {
41:     y=24;
42:     int a = objA.getX();
43: }
44:
45: //-----
46:
47: class C: public A
48: {
49:     protected:
50:         int z;
51:     public:
52:         int getZ();
53:         void setZ();
54: };
55:
56: int C::getZ()
57: {
58:     return z;
59: }
60:
61: void C::setZ()
62: {
63:     z=65;
64: }

```

Answer the following questions based on the code fragment given above:

- Is line 18 a valid access? Justify your answer.
- Is line 32 a valid statement? Justify your answer.
- Identify another invalid access statement in the code.
- Class C has `public` inheritance with the class A. Identify and list class C's `private`, `protected` and `public` member variables resulting from the inheritance.
- If class C had `protected` inheritance with the class A, identify and list class C's `private`, `protected` and `public` members variables resulting from the inheritance.

Question 2

Consider the class definitions below and answer the questions that follow:

```

class Date
{
public:
    friend ostream & operator<<(ostream & cout, const Date & d);
    Date(int y, int m, int d);
private:
    int year, month, day;
};

class Publication
{
public:
    Publication(const string & p, const Date & d,
        const string & t);
    Date GetDate( ) const;
    string GetPublisher( )const;
    string GetTitle( ) const;
private:
    string publisher;
    Date date;
    string title;
};

```

- (a) Implement the Date and the Publication classes.
- (b) Code the interface of a derived class Book for which the Publication class is the base class. The Book class has two additional member variables representing the ISBN number and the author of a book. Furthermore, the Book class contains member functions getISBN() and getAuthor() that return the ISBN number and the author respectively. The declaration must also include a constructor for the class Book.
- (c) Implement the Book class.
- (d) Recode the following interface such that class Magazine, derives from class Publication:

```

class Magazine
{
public:
    Magazine(const string & p, const Date & d, int ipy);
    int GetIssuesPerYear( ) const;
    Date getDate( ) const;
    string getPublisher( )const;
    string GetTitle( ) const;
private:
    int issuesPerYear;
    string publisher;
    Date date;
    string title;
};

```

- (e) Implement the Magazine class.
- (f) In a driver program embed code to do the following:

- (i) Declare an object `B` of type `Book`, with the following details:

publisher: FisherKing

date: 01/01/2000

title: Global Warming

isbn : 123456789

author: Ann Miller

- (ii) Output all the details of `Book B`.

- (iii) Declare an object `M` of type `Magazine`, with the following details:

publisher: Blue Marlin

date: 02/02/2005

title: The Earth and the Environment

number of issues per year: 12

- (iv) Output all the details of `Magazine M`.

- (v) Write a statement to overload `operator<<` as a friend function to the class `Book` and add the following implementation to your code:

```
ostream & operator<<(ostream & out, const Book & B)
{
    out<<B.title<<endl;
    out<<B.publisher<<endl;
    out<<B.date<<endl;
    out<<B.author<<endl;
    out<<B.ISBN<<endl;
}
```

You should obtain the following compiler errors:

```
In function `std::ostream& operator<<(std::ostream&,
                                     const Book&)':
error: `std::string Publication::title' is private
error: `std::string Publication::publisher' is private
error: `Date Publication::date' is private
```

Suggest two ways to fix this compiler problem.

Question 3

Write a function template for a function that has parameters for a partially filled array and for a value of the base type of the array. If the value is in the partially filled array, then the function returns the index of the first indexed variable that contains the value. If the value is not in the array, the function returns -1. The base type of the array is a type parameter. Notice that you

need two parameters to give the partially filled array: one for the array and one for the number of indexed variables used. Also write a suitable test program to test this function template.

Question 4

Write a template version of a search function for determining whether an array contains a particular value.

Question 5

Study the `Matrix` class interface and answer the questions that follow:

(Refer to the Notes at end of the question if you are unfamiliar with Matrices)

```
template<class Object>
class Matrix
{
public:
    Matrix( int row = 0, int col = 0 );
    void SetValue(Object value, int r, int c);
    Object GetValue( int r, int c) const;
    int GetRow() const;
    int GetCol() const;
    void OutPut(ostream & out) const;
private:
    vector< vector<Object> > array;
    int rows;
    int cols;
};
```

(a) Complete the implementation of the `Matrix` class where indicated:

```
template <class Object>
Matrix<Object>::Matrix (int row, int col)
{
    rows = row;
    cols = col;
    array.resize(row);
    for (int r = 0; r < row; r++)
        array[r].resize(col);
}
```

//SetValue assigns row r and column c of the Matrix to value

```
template <class Object>
void Matrix<Object>::SetValue(Object value, int r, int c)
{
    //Complete code here
}
```

```

//GetValue returns the value in row r and col c of the Matrix
template <class Object>
Object Matrix<Object>::GetValue( int r, int c) const
{
    //Complete code here
}

//GetRow returns rows
template<class Object>
int Matrix<Object>::GetRow() const
{
    //Complete code here
}

//GetCol returns cols
template<class Object>
int Matrix<Object>::GetCol() const
{
    //Complete code here
}

//Outputs the matrix in a tabular format (see Notes for example)
template <class Object>
void Matrix<Object>::OutPut(ostream & out) const
{
    //Complete code here
}

//Operator+ is overloaded as a non-friend, non-member function. This
//function adds two Matrices (see Notes for example)
template<class Object>
Matrix<Object> operator+(const Matrix<Object> & x, const
Matrix<Object> & y)
{
    //Complete code here
}

```

b) Test your implementation by coding a main function to perform the following:

(i) Declare three, 2 by 2 integer matrices, M1, M2, and M3;

(ii) Store the following values in M1:

1 2

3 4

(iii) Store the following values in M2:

5 6

7 8

(iv) Store the sum of M1 and M2 in M3 using operator+.

- (v) Output all three matrices.

Notes:

In mathematics, a matrix (plural matrices) is a rectangular table of numbers or, more generally, a table consisting of abstract quantities that can be added and multiplied. For example, a 4 by 3 matrix is represented as:

6	6	6
5	3	2
3	1	2
2	7	9

Two matrices can be added if, and only if, they have the same dimensions. (That is, both matrices have matching numbers of rows and columns.) We define their sum by constructing a third matrix whose entries are the sum of the corresponding entries of the original two matrices. For example:

4	3	4		2	2	1		6	5	5
1	2	3	+	1	3	2	=	2	5	5
2	2	1		3	4	5		5	6	6

Question 6

Write a program that inputs two C string variables, `first` and `last`, each of which the user should enter with his or her name. First, convert both C strings to lowercase. Your program should then create a new C string that contains the full name in pig latin with the first letter capitalized for the first and last name. The rules to convert a word into pig latin are as follows:

If the first letter is a consonant, move it to the end and add "ay" to the end.

If the first letter is a vowel, add "way" to the end.

For example, if the user inputs "Erin" for the first name and "Jones" for the last name, then the program should create a new string with the text "Erinway Onesjay" and print it.

Question 7

- (a) Write a sorting function that is similar to Display 7.12 in Chapter 7 in Savitch, except that it has an argument for a vector of `ints` rather than an array. This function will not need a parameter like `number_used` as in Display 7.12, since a vector can determine the number used with the member function `size()`. This sort function will have only this one parameter, which will be of a vector type. Use the selection sort algorithm (which was used in Display 7.12).
- (b) Write a program that reads in a list of integers into a vector with base type `int`. Provide the facility to either read this vector from the keyboard or from a file, at the user's option. If the user chooses file input, the program should request a file name. The output is to be a two-column list. The first column is a list of the distinct vector elements; the second column is a count of the number of occurrences of each element. The list should be sorted

on entries in the first column, largest to smallest. Adapt the sorting function from (a) as necessary.

For example, for the input

-12 3 -12 4 1 1 -12 1 -1 1 2 3 4 2 3 -12

The output should be

N	Count
4	2
3	3
2	2
1	4
-1	1
-12	4

Question 8

Write a recursive function that returns the sum of the integers between any two integer numbers inclusive. For example if we want to calculate the sum of integers between the integer numbers 13 and 17 then the sum will be $13 + 14 + 15 + 16 + 17 = 75$. This recursive function will expect two integer parameters and will return a double.

9 EXAMINATION

A 2 hour examination will be scheduled for this module. Please refer to the *my Studies @ Unisa* brochure for general examination guidelines and examination preparation guidelines.

10 FREQUENTLY ASKED QUESTIONS

What if I cannot find the prescribed book?

Do not contact the lecturers if you have problems obtaining the textbook. If you have any difficulties with obtaining books from the official booksellers bookshops, please contact vospresc@unisa.ac.za.

You can also buy an e-book version of Savitch at www.coursesmart.com.

What if I fail to submit my assignment on time?

A grace period is allowed for submission difficulties via MyUnisa. If the MyUnisa system is down when you try to submit an assignment, do not contact the lecturers. Wait until the problem has been solved and submit as soon as possible. We are usually aware of the problems with MyUnisa. If you submit late for any other reason, include a note with the assignment with the reasons for the late submission. No assignment will be marked (i.e. a mark of ZERO will be awarded) after the solutions for the particular assignment have been published. This is usually a week or two after the due date.

How do I request an extension? (Do not!)

Please do not phone, fax or email for an extension. Submit the assignment as soon as possible, and include a note of explanation.

What if there are mistakes in the marking of assignments? (Do NOT re-submit to Assignments Department!)

We use a team of external markers that are sub-contracted for the purpose of marking assignments during the year. There are close to 500 students enrolled for the module. For this reason, inconsistency in the marking style of individual markers may be encountered. We request that students only query assignment marking where the marks will change significantly (i.e. more than 5%). Please follow the RE-MAIL PROCEDURE below should you require that an assignment be remarked. If your marks are added incorrectly or a question is not marked which we stated (in the solutions) was to be marked, or you feel strongly that you were penalised unfairly, follow the procedure below. If you phone we will just tell you to follow the RE-MAIL PROCEDURE.

The RE-MAIL procedure:

Scan your marked assignment and attach it to an e-mail, addressed to the module email, TOGETHER WITH A MESSAGE stating your marking dilemma with the specified questions of the assignment.

What if I don't receive my study material or I lose it?

All study material is downloadable from myUnisa web site for COS1512. Please download electronic copies (PDF files) of tutorial letters and the study guide from Official Study Material, and the CD from Additional Resources.

May I send my assignment by email?

No. Assignments have to be registered. Students may submit assignments either by post or Mobile MCQ submission or electronically via myUnisa. Assignments are not accepted via fax or e-mail.

Have we received your assignment?

If you want to find out whether an assignment has been received by Unisa, marked or returned, look at the status of your assignment on MyUnisa.

11 CONCLUSION

Do not hesitate to contact your lecturer or e-tutor by email if you are experiencing problems with the content of this tutorial letter or any aspect of the module.

I wish you a fascinating and satisfying journey through the learning material and trust that you will complete the module successfully.

Enjoy the journey!

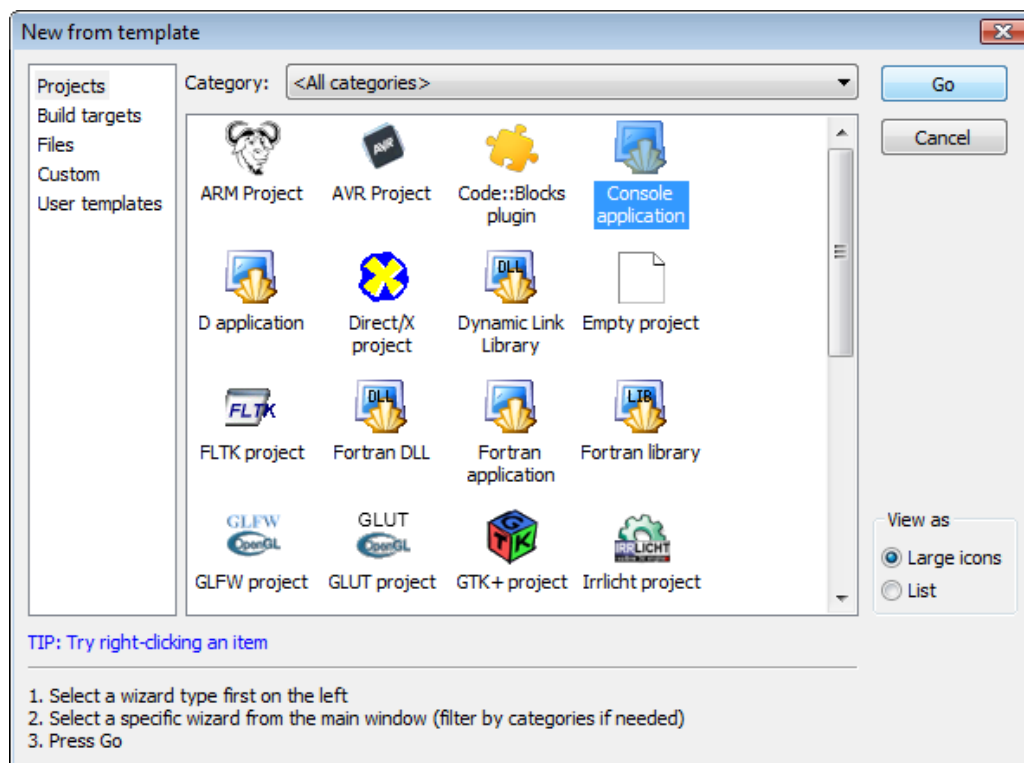
COS1512 Team

12 APPENDIX A: THE SOFTWARE FOR COS1512

1. Creating a project

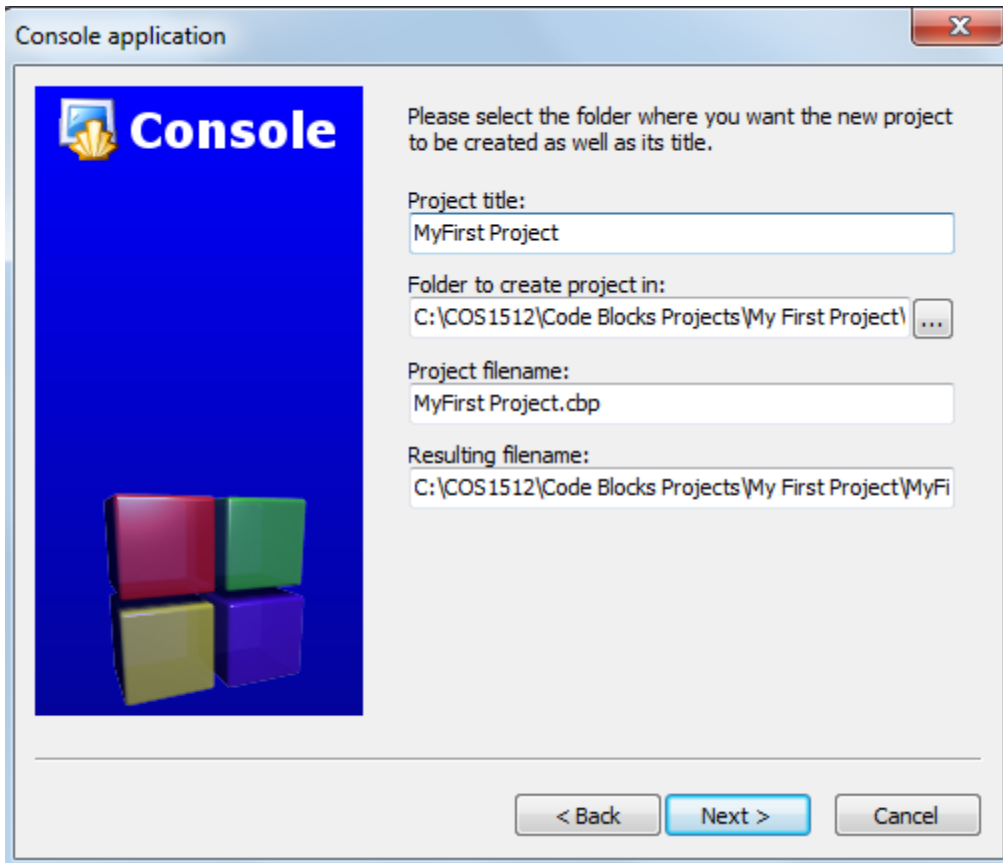
Starting a new project

Launch the Project Wizard through *File->New->Project*. From the pre-configured templates for various types of projects, select **Console application** and click **Go**.

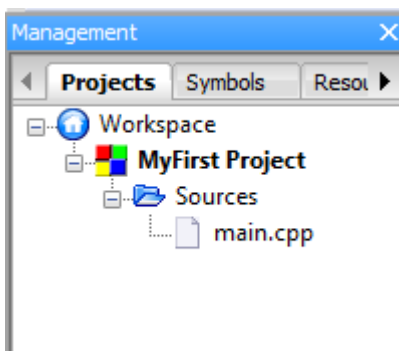


The console application wizard will appear next. Continue through the menus, selecting **C++** when prompted for a language. In the next screen, give the project a **name** and

type or select a destination folder. As seen below, Code::Blocks will generate the remaining entries from these two.



Finally, the wizard will ask if this project should use the default compiler (normally GCC) and the two default builds: **Debug** and **Release**. All of these settings are fine. Press finish and the project will be generated. The main window will turn gray, but that is not a problem, the source file needs only to be opened. In the **Projects** tab of the **Management** panel on the left expand the folders and double click on the source file **main.cpp** to open it in the editor.



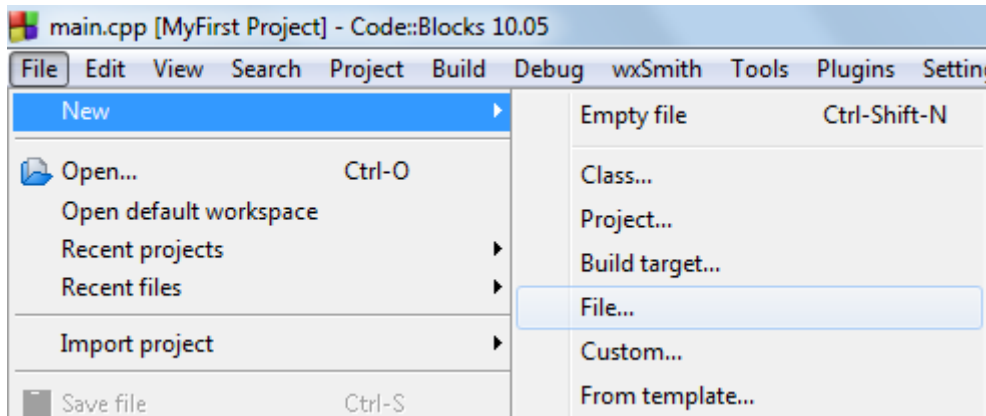
This file contains the following standard code.

main.cpp

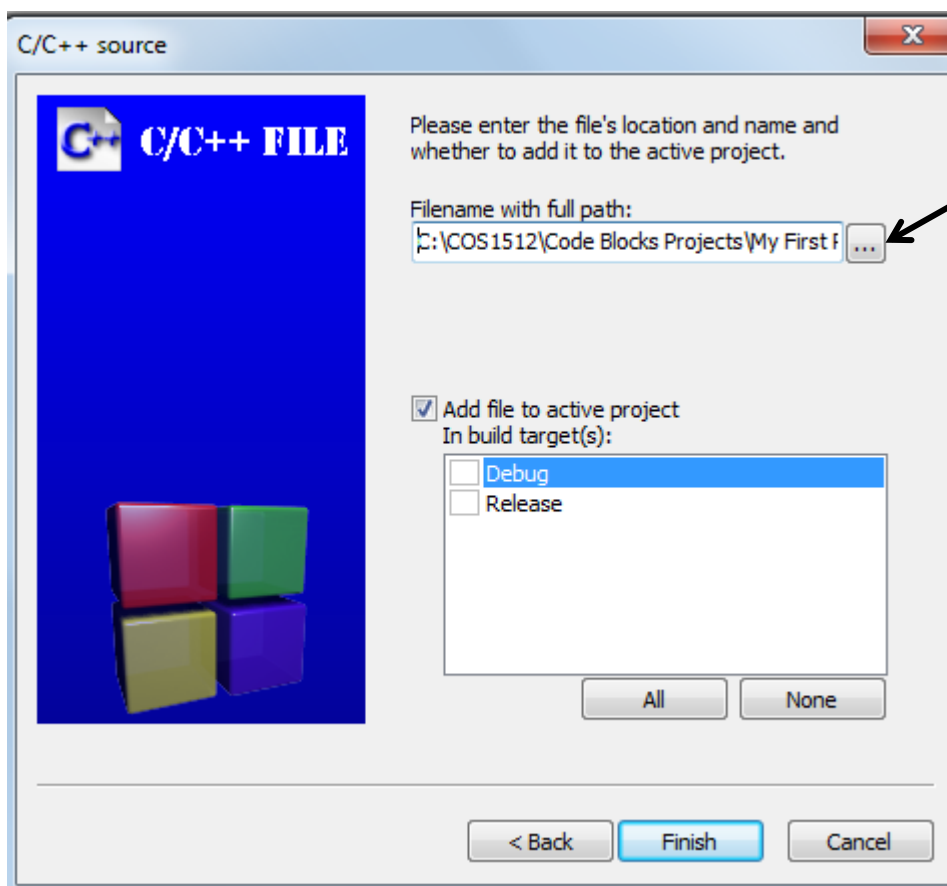
```
1. #include <iostream>
2.
3. using namespace std;
4.
5. int main()
6. {
7.     cout<<"Hello world!"<<endl;
8.     return 0;
9. }
```


Adding a file to your project

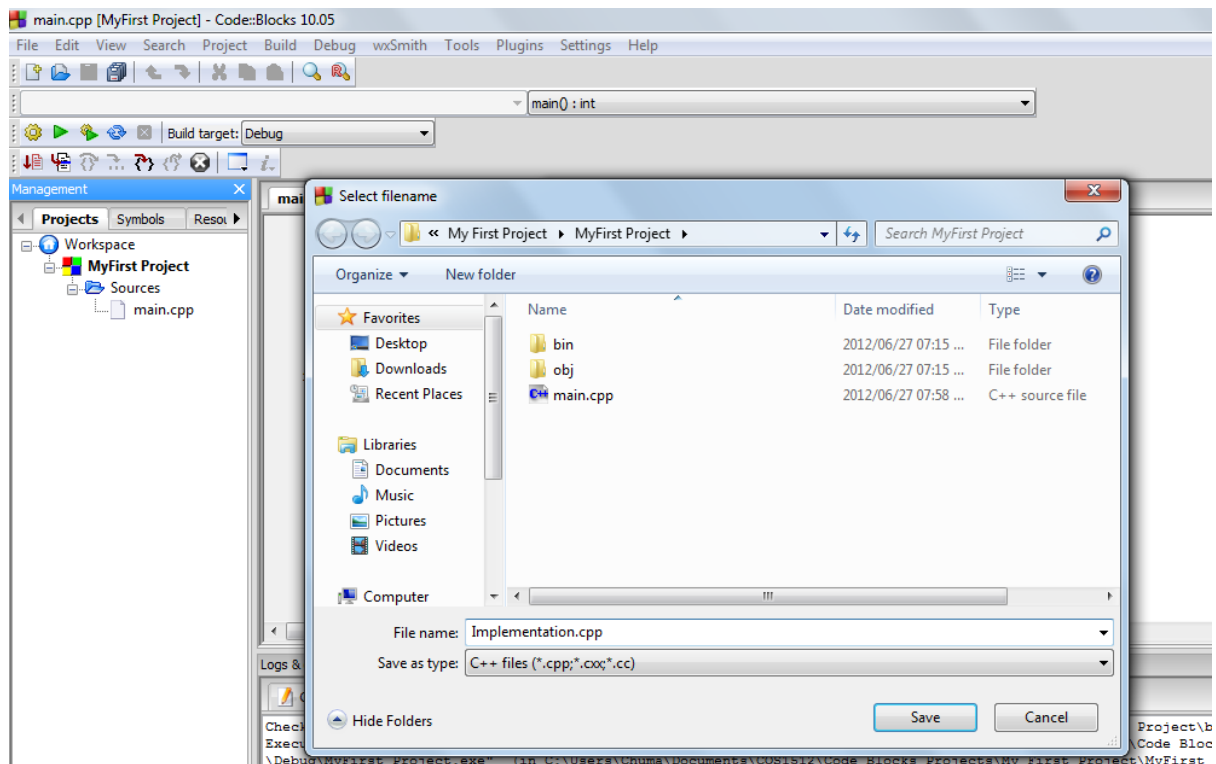
To add the new file to the project, bring up the file template wizard through either *File->New->File...* or *Main Toolbar->New file (button)->File...*



Select **C/C++ source** and click **Go**. Continue through the menus, same as what you have done before. The last menu will present you with several options. Enter the new filename and location (as noted, the full path is required). You can browse for the file by clicking the browse button (see below) to display the file browser window to save the file's location. Checking **Add file to active project** will store the filename in the **Sources** folder of the **Projects** tab of the **Management** panel. Checking any of the build targets will alert Code::Blocks that the file should be compiled and linked into the selected target(s). click **Finish** to generate the file.



The newly created file should open automatically; if it does not, open it by double clicking on its file in the **Projects** tab of the **Management** panel. You can now add code to the new file. Be careful not to save your files with `.c` extension (this is not a C++ extension).



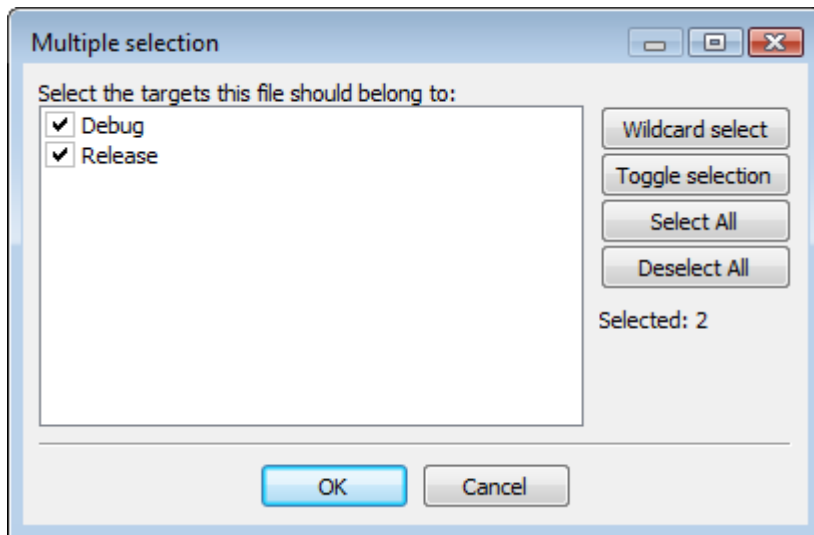
Adding a pre-existing file

Copy an existing file to your project folder or launch a plain text editor (for example Notepad), and add the following code.

Interface.h

```
1. #ifndef INTERFACE_H_INCLUDED
2. #define INTERFACE_H_INCLUDED
3.
4. void hello();
5.
6. #endif // INTERFACE_H_INCLUDED
```

Save this file as a header (**Interface.h**) in the same directory as the other source files in this project. Back in Code::Blocks, click *Project->Add files...* to open a file browser. Here you may select one or multiple files (using combinations of *Ctrl* and *Shift*). (The option *Project->Add files recursively...* will search through all the subdirectories in the given folder, selecting the relevant files for inclusion.) Select **Interface.h**, and click **Open** to bring up a dialog requesting to which build targets the file(s) should belong. For this example, select both targets.



Note: if the current project has only one build target, this dialog will be skipped.

Returning to the main source (**main.cpp**) include the header file and replace the `cout` function to match the new setup of the project.

main.cpp

```
1. #include "Interface.h"
2.
3. int main()
4. {
5.     hello();
6.     return 0;
7. }
```

Press **Ctrl-F9** or **File->Build**, or **Compiler Toolbar->Build** (button - the gear) to compile the project. If the following output is generated in the build log (in the bottom panel) then all steps were followed correctly.

----- Build: Debug in MyFirst Project -----

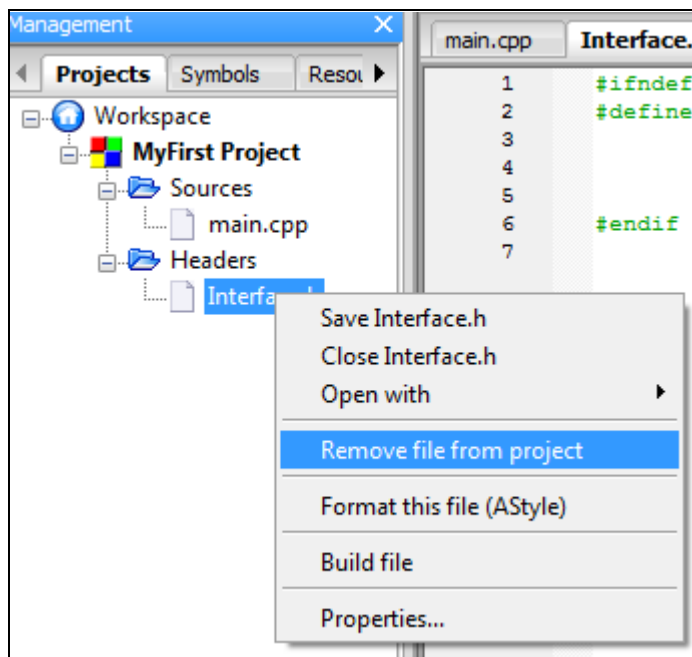
```
Compiling: main.cpp
Linking console executable: bin\Debug\MyFirst Project.exe
Output size is 913.10 KB
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings
```

You can now “run” the project by either clicking the *Run* button or hitting **Ctrl-F10**.

Note: the option F9 (for build and run) combines these commands, and may be more useful in some situations.

Removing a file

You can remove a file by simply right-clicking on the file name in the **Projects** tab of the **Management** panel and selecting **Remove file from project**.



*Note: removing a file from a project does **not** physically delete it.*

2. Printing

To print a C++ program, choose the "Print" option on the "File" menu of Code::Blocks. (If you are submitting an assignment via *myUnisa*, you don't need to print. Simply paste the code of your program into a word processor file.)

Printing the *output* of a program is somewhat trickier. There are (at least) two ways to print the output of a text-based program (a console application):

Method 1

To print the text from the I/O window after running your program, you can copy the text to a word processor (an editor). The steps involved are as follows:

- Position the mouse over the console window (the output window). Right-click, and choose Mark from the drop-down menu.
- Hold the Shift key down and use the arrow keys to mark (highlight) the text as a block.
- Press the Enter key to copy the highlighted text to the clipboard.
- You can now paste it in a word processor (editor) of your choice and print it.

Method 2

Sometimes the above method can be somewhat laborious and problematic, especially if there is so much output that it scrolls off the top of the screen. In this case, you can send the output directly to the printer (while the program is running) like this:


- Run your program, and when it has finished executing (and you are happy with the output) close the console window.
- Open a separate DOS window (or Command window) and change the directory to where your program is. (You'll need to type something like **cd \unisa\cos1512.**)

- Test whether your program is actually in the current directory by typing its name at the DOS prompt, eg. **first.exe** followed by <Enter>. If you get the message "Bad command or filename" you are either in the wrong directory or the name of the executable file is incorrect. You must be able to run the program from the DOS prompt before proceeding.
- Make sure that your printer is switched on, is "On-line" and has paper in it, etc.
- Press <Ctrl+P> to ensure that all the output generated from now on is sent to the printer.
- Type the name of the executable file, eg. **first.exe** and press <Enter> to run your program again.
- Enter any values that the program requires as input.
- When the program terminates, press <Ctrl+P> again to turn off the printing mode. All the output (and input) of the program should have been sent to the printer.
- Now you can close the console window.

Unfortunately, this method won't help if you intend submitting your assignment via *myUnisa*. You'll have to use Method 1.

13 APPENDIX B: SOURCE LISTINGS OF THE SAMPLE PROGRAMS IN THE TEXTBOOK

The source code listings of the sample programs in the 8th and 9th edition of the text book can be found at the Companion Website for the text book. Go to www.pearsinternationalEditions.com/savitch, select the text book and then click on **Companion Website**. Click on the **Register** button, and type in the student access code found beneath the pull tab of the **ONLINE ACCESS** card in front of the text book.

The source code listings of the sample programs in the 7th edition of the text book can be found at <http://www.aw.com/cssupport>. Click on  under “Author Search”, and then on “Savitch” in the resulting list of authors. Follow the link provided for the text book. Now double-click on “PSCPP6e-SourceCode.zip” and click on Open. A list of folders, one for each chapter, will be displayed. Each folder contains the source listings for Displays in the chapter. Display 9.06 for example will be listed as “09-06.cpp”. If you double-click on the file, the source listing will open up in Code::Blocks.

14 APPENDIX C: GLOSSARY

The following link provides a glossary for English/Afrikaans IT terminology:

<http://www.coetzee.org/woordelys/>

15 APPENDIX D: Solution to assignment 3

Question 1

For this question you had to answer questions based on the following code fragment:

```

1:  #include <iostream>
2:  using namespace std;
3:
4:  //-----
5:
6:  class A
7:  {
8:  private:
9:      int x;
10: protected:
11:     int getX();
12: public:
13:     void setX();
14: };
15:
16: int A::getX()
17: {
18:     return x;
19: }
20:
21: void A::setX()
22: {
23:     x=10;
24: }
25:
26: //-----
27: class B
28: {
29: private:
30:     int y;
31: protected:
32:     A objA;
33:     int getY();
34: public:
35:     void setY();
37: };
38:
39: void B::setY()
40: {
41:     y=24;
42:     int a=objA.getX();
43: }
44:
45: //-----
46:
47: class C: public A
48: {
49:     protected:

```

```

50:         int z;
51:     public:
52:         int getZ();
53:         void setZ();
54:     };
55:
56:     int C::getZ()
57:     {
58:         return z;
59:     }
60:
61:     void C::setZ()
62:     {
63:         z=65;
64:     }

```

Answer the following questions based on the code fragment given above:

- (a) Is line 18 a valid access? Justify your answer?

Yes.

The variable `x` is a private data member of the class `A` and therefore it can only be accessed by other member functions and operators of the class `A`. `getX()` is a member function of class `A` and therefore line 18 is a valid access.

- (b) Is line 32 a valid statement? Justify your answer.

Yes.

An object of class `A` has been included as a protected data member of class `B`.

- (c) Identify another invalid access statement in the code.

Line 42 (`int a=objA.getX();`) is invalid.

`getX()` is a protected member function of class `A` and can therefore only be accessed by other member functions and operators of the class `A` and by classes derived from class `A`.

- (d) Class `C` has public inheritance with the class `A`. Identify and list class `C`'s private, protected and public data members resulting from the inheritance.

With public inheritance, the public and protected members of the base class `A` are inherited as public and protected members of the derived class `C`.

Private data members or member functions resulting from the inheritance: None

Protected data members or member functions resulting from the inheritance: `getX()`

Public data members or member functions resulting from the inheritance: `setX()`

- (a) If class `C` had protected inheritance with the class `A`, identify and list class `C`'s private, protected and public data members resulting from the inheritance.

With protected inheritance, public and protected members of the base class become protected members of the derived class.

Private data members or member functions resulting from the inheritance: None

Protected data members or member functions resulting from the inheritance: setX() and getX()

Public data members or member functions resulting from the inheritance: None

Discussion:

When deriving a class from a public base class, public members of the base class become public members of the derived class and protected members of the base class become protected members of the derived class. A base class' private members are never directly accessible from a derived class, but can be accessed through calls to the public and protected members of the base class.

When deriving from a protected base class, public and protected members of the base class become protected members of the derived class. When deriving from a private base class, public and protected members of the base class become private members of the derived class. Private and protected inheritance are not "is-a" relationships [Reference: Study Guide Appendix D, chapter 15].

Question 2

For this question you had to answer questions based on the following code fragment:

```
class Date
{
public:
    friend ostream & operator<<(ostream & cout, const Date & d);
    Date(int y, int m, int d);
private:
    int year, month, day;
};

class Publication
{
public:
    Publication(const string & p, const Date & d,
               const string & t);
    Date GetDate( ) const;
    string GetPublisher( )const;
    string GetTitle() const;
private:
    string publisher;
    Date date;
    string title;
};
```

- (a) Implement the Date and the Publication classes.

File Name: Date.cpp

```
#include <iostream>
#include "Date.h"

using namespace std;

Date::Date(int y, int m, int d): year(y), month(m), day(d){}
ostream & operator<<(ostream & out, const Date & d)
{
    out<<d.day<<"/"<<d.month<<"/"<<d.year<<endl;
}
```

File Name: Publication.cpp

```
#include <iostream>
#include <string>
#include "Date.h"
#include "Publication.h"

using namespace std;

Publication::Publication( const string & p, const Date & d,
                        const string & t): publisher(p),date(d),title(t)
{ }

Date Publication::GetDate() const
{
    return date;
}

string Publication::GetPublisher()const
{
    return publisher;
}

string Publication::GetTitle() const
{
    return title;
}
```

- (b) Code the interface of a derived class Book for which the Publication class is the base class. The Book class has two additional data members representing the ISBN number and the author of a book. Furthermore, the Book class contains member functions getISBN() and getAuthor() that returns the ISBN number and the author respectively. The declaration must also include a constructor for the class Book.

```
#ifndef BOOK_H
#define BOOK_H
#include "Publication.h"
#include "Date.h"
#include <string>
```

```

using namespace std;

class Book: public Publication
{
public:
    Book(const string & p, const Date & d, const string & t,
          const string & auth, const string & isbn);
    string getAuthor()const;
    string getISBN()const;
private:
    string ISBN;
    string author;

};

#endif

```

(c) Implement the Book class.

```

#include "Book.h"
#include <string>
using namespace std;
Book::Book(const string & p, const Date & d, const string & t,
            const string & auth, const string & isbn):
    Publication(p,d,t),
    ISBN(isbn), author(auth){}

string Book::getAuthor()const
{
    return author;
}

string Book::getISBN() const
{
    return ISBN;
}

```

Note, the constructors of the Book class call the Publication constructor (shown in bold) in their initializer lists. Constructing a derived class object by first constructing its inherited portion is a standard practice. (For a complete discussion on this subject - refer to pages 851 -853 (8th ed) of Savitch / pages 865-868 (7th ed) of Savitch, under the section entitled "Constructors in Derived Classes").

(d) Recode the following interface such that class Magazine, derives from class Publication:

```

#ifndef MAGAZINE_H
#define MAGAZINE_H
#include "Publication.h"
class Magazine: public Publication
{

```

```

public:
Magazine(const string & p, const Date & d, const string & t,
        int ipy);
int GetIssuesPerYear( ) const;

private:
int issuesPerYear;
};
#endif

```

Note how the class requires less code due to inheritance.

- (e) Implement the Magazine class.

```

#include "Magazine.h"
Magazine::Magazine(const string & p, const Date & d,
const string & t, int ipy):Publication(p,d,t), issuesPerYear(ipy)
{}
int Magazine::GetIssuesPerYear()const
{
return issuesPerYear;
}

```

- (f) For this question you had to design a driver program to test your classes:

```

#include <iostream>
#include "Date.h"
#include "Publication.h"
#include "Book.h"
#include "Magazine.h"

using namespace std;
int main()
{
    Date date1(2000,1,1);
    Date date2(2005,2,2);

    Book B("FisherKing", date1, "Global Warming", "123456789",
    "Ann Miller");
    cout<<B.GetTitle()<<endl;
    cout<<B.GetPublisher()<<endl;
    cout<<B.GetDate()<<endl;
    cout<<B.getAuthor()<<endl;
    cout<<B.getISBN()<<endl;
    Magazine M("Blue Marlin", date2,
    "TheEarth and the Environment",12);
    cout<<M.GetTitle()<<endl;
    cout<<M.GetPublisher()<<endl;
    cout<<M.GetDate()<<endl;
    cout<<M.GetIssuesPerYear()<<endl;

    return 0;
}

```

- (g) Write a statement to overload operator<< as a friend function to the class Book and insert the following implementation to your code:

```
ostream & operator<<(ostream & out, const Book & B)
{
    out<<B.title<<endl;
    out<<B.publisher<<endl;
    out<<B.date<<endl;
    out<<B.author<<endl;
    out<<B.ISBN<<endl;
}
```

You should obtain the following compiler errors:

```
In function `std::ostream& operator<<(std::ostream&,
                                     const Book&)':
error: `std::string Publication::title' is private
error: `std::string Publication::publisher' is private
error: `Date Publication::date' is private
```

Suggest two ways to fix this compiler problem.

Method 1: Use the accessor functions as shown below:

```
ostream & operator<<(ostream & out, const Book & B)
{
    out<<B.GetTitle()<<endl;
    out<<B.GetPublisher()<<endl;
    out<<B.GetDate()<<endl;
    out<<B.getAuthor()<<endl;
    out<<B.getISBN()<<endl;
}
```

Method 2: Change the member variables of Publication into protected access. As they are now protected and not private they become accessible to the derived class Book. The protected members of Publication become protected members of Book. They now can be directly accessed by member functions and friend functions of Book.

```
class Publication
{
public:
    Publication(const string & p, const Date & d,
               const string & t);
    Date GetDate() const;
    string GetPublisher() const;
    string GetTitle() const;
protected:
    string publisher;
    Date date;
    string title;
};
```

Full Programming Listing:

File name: Date.h

```
#ifndef DATE_H
#define DATE_H
#include <iostream>

using namespace std;
class Date
{
public:
    friend ostream & operator<<(ostream & cout, const Date & d);
    Date(int y, int m, int d);
private:
    int year, month, day;
};

#endif
```

File name: Date.cpp

```
#include <iostream>
#include "Date.h"

using namespace std;

Date::Date(int y, int m, int d): year(y), month(m), day(d){}
ostream & operator<<(ostream & out, const Date & d)
{
    out<<d.day<<"/"<<d.month<<"/"<<d.year<<endl;
}
```

File name: Publication.h

```
#ifndef PUBLICATION_H
#define PUBLICATION_H
#include <string>
#include "Date.h"

using namespace std;
class Publication
{
public:
    Publication(const string & p, const Date & d, const string & t);
    Date GetDate( ) const;
    string GetPublisher( )const;
    string GetTitle() const;
private:
    string publisher;
    Date date;
    string title;
};

#endif
```

File name:Publication.cpp

```

#include <iostream>
#include <string>
#include "Date.h"
#include "Publication.h"
using namespace std;

string Publication::GetTitle() const
{
    return title;
}
Publication::Publication( const string & p, const Date & d, const
string & t): publisher(p),date(d),title(t){}

Date Publication::GetDate() const
{
    return date;
}

string Publication::GetPublisher()const
{
    return publisher;
}

```

Filename: Book.h

```

#ifndef BOOK_H
#define BOOK_H
#include "Publication.h"
#include "Date.h"
#include <string>

using namespace std;

class Book: public Publication
{
public:
    friend ostream & operator << (ostream & out, const Book & B);
    Book(const string & p, const Date & d, const string & t,
        const string & auth,const string & isbn);
    string getAuthor()const;
    string getISBN()const;
private:
    string ISBN;
    string author;
};

#endif

```

Filename: Book.cpp

```
#include "Book.h"
#include <string>
using namespace std;

Book::Book(const string & p, const Date & d, const string & t,
           const string & auth, const string & isbn):
    Publication(p,d,t),
    ISBN(isbn), author(auth){}

string Book::getAuthor()const
{
    return author;
}

string Book::getISBN() const
{
    return ISBN;
}

ostream & operator<<(ostream & out, const Book & B)
{
    out<<B.GetTitle()<<endl;
    out<<B.GetPublisher()<<endl;
    out<<B.GetDate()<<endl;
    out<<B.getAuthor()<<endl;
    out<<B.getISBN()<<endl;
}
```

Filename: Magazine.h

```
#ifndef MAGAZINE_H
#define MAGAZINE_H

#include "Publication.h"
class Magazine:public Publication
{
public:
    Magazine(const string & p, const Date & d, const string & t,
             int ipy);
    int GetIssuesPerYear( ) const;
private:
    int issuesPerYear;
};
#endif
```

Filename: Magazine.cpp

```
#include "Magazine.h"

Magazine::Magazine(const string & p, const Date & d, const string & t,
int ipy):Publication(p,d,t), issuesPerYear(ipy)
{}
```



```

int Magazine::GetIssuesPerYear()const
{
    return issuesPerYear;
}

Filename: Test.cpp
#include <iostream>
#include "Date.h"
#include "Publication.h"
#include "Book.h"
#include "Magazine.h"

using namespace std;
int main()
{
    Date date1(2000,1,1);
    Date date2(2005,2,2);

    Book B("FisherKing", date1, "Global Warming", "123456789",
        "Ann Miller");
    cout<<B.GetTitle()<<endl;
    cout<<B.GetPublisher()<<endl;
    cout<<B.GetDate()<<endl;
    cout<<B.getAuthor()<<endl;
    cout<<B.getISBN()<<endl;
    Magazine M("Blue Marlin", date2,
        "TheEarth and the Environment",12);
    cout<<M.GetTitle()<<endl;
    cout<<M.GetPublisher()<<endl;
    cout<<M.GetDate()<<endl;
    cout<<M.GetIssuesPerYear()<<endl;

    return 0;
}

```

Question 3

Discussion:

For this question, you had to define a function template that searches an array for a specific value and returns the index of the first occurrence of that value. The template should have parameters for a partially filled array and for a value of the base type of the array. If the value is in the partially filled array, then the function returns the index of the first occurrence of that value, otherwise the function returns -1. The base type of the array is a type parameter.

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one variable type or class without repeating the code for each type. For instance, with this program we used the same `search()` function for an array of doubles (`Doubles`), an array of characters (`Characters`) and an array of integers (`Integers`). Without templates we would have had to write a separate `search()` function for each type of array. (Section 17.1 of Savitch has detailed explanations on the declarations of function templates.)

A template parameter is a special kind of parameter that can be used to pass a type as a parameter. The function templates can use these parameters as if they were regular types. The declaration for the `search()` function combined template parameters with a defined parameter type:

```
template<class T>

int search(T array[], int n, T target)
```

Be cautious when using template parameters - you cannot apply it without considering all the implications.

For example:

```
template<class T>

int search( T array[], T n, T target)
```

☞ This does not make sense as we want to run a loop `n` number of times to search the array for our target element - `n` must certainly be of type integer. Hence it does not make sense within this context to declare `n` as a template parameter.

Program Listing:

```
#include <iostream>

//Precondition: the array base type must have operator== defined,
//&& n <= declared size of the array argument.
//Postcondition: Function returns index of the first
//occurrence of target in array. If target is not in the
//array, the function returns -1

using namespace std;

template<class T>
int search( T array[], int n, T target)
{
    for ( int i = 0; i < n; i++ )
    {
        if (array[i] == target)
            return i;
    }
    return -1;
}

int main(){
    char Characters[14] = { 'M', 'A', 'P', 'C', 'E' };
    int Integers[14] = { 1, 4, 3, 5, 3, 6, 8, 9, 10, 7 };
    double Doubles[14] = {2.99,8.77,4.88,6.44,3.45};

    cout << " C found at index "
         << search( Characters,5,'C')
         << " in array of characters"
         << endl;
    cout << " c found at index "
```

```

        << search( Characters,5,'c')
        << " in array of characters"
        << endl;
    cout << " 5 found at index "
        << search( Integers, 10, 5)
        << " in array of integers"
        << endl;
    cout << " 3.45 found at index "
        << search( Doubles, 10, 3.45)
        << " in array of doubles"
        << endl;
}

```

Output:

```

C found at index 3 in array of characters
c found at index -1 in array of characters
5 found at index 3 in array of integers
3.45 found at index 4 in array of doubles
Press any key to continue . . .

```

Question 4

Discussion:

For this question, you had to specify a template version of a search function to determine whether an array contains a particular value. Here only two of the parameters had to be template type parameters, namely the array type and the target (i.e. the element being searched). As discussed, with the previous question, it does not make sense to alter the other parameters into template type parameters. We also included a templated `Output()` function to output an array of any type and to call the `search()` algorithm to further promote reuse.

Program Listing:

```

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

//Precondition: the array base type must have operator== defined,
//&&    n <= declared size of the array argument.
//Postcondition: Function returns true at the first
//occurrence of target in array. If target is not in the
//array, the function returns false

template<class T>
bool search( T array[], int n, T target)
{
    for ( int i = 0; i < n; i++ )
    {
        if (array[i] == target)
            return true;
    }
    return false;
}

```

```

}
//Precondition: the array base type must have operator<< defined,
//&&    n <= declared size of the array argument.

template <class T>
void Output(T array[], int n, T target, string s)
{
    if (search(array,n,target))
        cout << s << "does contain " << target <<endl;
    else
        cout << s << "does not contain " << target <<endl;
}

int main()
{
    char Characters[14] = { 'M', 'A', 'P', 'C', 'E' };
    int Integers[14] = { 1, 4, 3, 5, 3, 6, 8, 9, 10, 7 };
    double Doubles[14] = {2.99,8.77,4.88,6.44,3.45};

    Output( Characters,5,'C', "Array Characters ");
    Output( Characters,5,'c', "Array Characters ");
    Output( Integers, 10, 5 , "Array Integers ");
    Output( Integers, 10, 15 , "Array Integers ");
    Output( Doubles, 10, 3.45, "Array Doubles ");
    Output( Doubles, 10, 3.455, "Array Doubles ");
}

```

Output:

```

Array Characters does contain C
Array Characters does not contain c
Array Integers does contain 5
Array Integers does not contain 15
Array Doubles does contain 3.45
Array Doubles does not contain 3.455
Press any key to continue . . .

```

Question 5

For this question, you had to implement the operations of a `Matrix` class template.

(a) The code to be inserted shown in bold.

Discussion:

Until now, we have placed the definition of the class (in the header file) and the definition of the member functions (in the implementation file) in separate files. However this does not work with class templates. As shown below both the interface and implementation are within the header file `Matrix.h`

File name: **Matrix.h**

```

#ifndef MATRIX_H
#define MATRIX_H

```

```

#include <iostream>
#include <vector>
#include <cassert>
using namespace std;
template<class Object>
class Matrix
{
public:
    Matrix( int r = 0, int c = 0 );
    void SetValue(Object value, int row, int col);
    Object GetValue( int row, int col) const;
    int GetRow() const;
    int GetCol() const;
    void OutPut(ostream & out) const;
private:
    vector< vector<Object> > array;
    int rows;
    int cols;
};

template <class Object>
Matrix<Object>::Matrix (int row, int col)
{
    rows = row;
    cols = col;
    array.resize(row);
    for (int r = 0; r < row; r++)
        array[r].resize(col);
}

template <class Object>
void Matrix<Object>::SetValue(Object value, int row, int col)
{
    array[row][col] = value;
}

template <class Object>
Object Matrix<Object>::GetValue(int row, int col) const
{
    return array[row][col];
}

template<class Object>
int Matrix<Object>::GetRow() const
{
    return rows;
}

template<class Object>
int Matrix<Object>::GetCol() const
{
    return cols;
}

```

```

}

template <class Object>
void Matrix<Object>::OutPut(ostream & out) const
{
    for (int r = 0; r < rows; r++)
    {
        for(int c = 0; c < cols; c++)
        {
            out<<array[r][c]<<'\t';
        }
        cout<<endl;
    }
}

template<class Object>
Matrix<Object> operator+(const Matrix<Object> & x,
                        const Matrix<Object> & y)
{
    int xrow = x.GetRow();
    int xcol = y.GetCol();
    assert( xrow == y.GetRow() && xcol == y.GetCol());
    Matrix<Object> temp(xrow,xcol);
    for (int r = 0; r < xrow; r++)
    {
        for(int c = 0; c < xcol; c++)
        {
            Object sum = x.GetValue(r,c) + y.GetValue(r,c);
            temp.SetValue(sum, r, c);
        }
    }
    return temp;
}

#endif

```

(b) For this question you had to write a test program for your `Matrix` class:

```

#include <iostream>
#include "Matrix.h"
using namespace std;

int main()
{
    Matrix<int> M1(2,2);
    Matrix<int> M2(2,2);

    M1.SetValue(1,0,0);
    M1.SetValue(2,0,1);
    M1.SetValue(3,1,0);
    M1.SetValue(4,1,1);

    M2.SetValue(5,0,0);
    M2.SetValue(6,0,1);
}

```

```

M2.SetValue(7,1,0);
M2.SetValue(8,1,1);

Matrix<int> M3(2,2);
M3 = M1 + M2;

M1.Output(cout); cout<<endl;
M2.Output(cout);cout<<endl;
M3.Output(cout);cout<<endl;

return 0;
}

```

Output Produced:

```

1      2
3      4

5      6
7      8

6      8
10     12

```

Press any key to continue . . .

Note:

As shown above - we also have the possibility to write class templates, so that a class can have members that use template parameters as types. C++ class templates are used where we have multiple copies of code for different data types with the same logic. If a set of functions or classes have the same functionality for different data types, they become good candidates for being written as templates. C++ class templates are ideal for container classes (a class of objects that is intended to contain other objects). Examples of container classes will be the STL classes like `vector` (chapter 8, Savitch), and `list`. Once the code is written as a C++ class template, it can support all data types. (See section 17.2, Savitch for full explanations on Class Templates.) For instance the `Matrix` template above can accommodate a `Matrix` of any type, be it `strings`, `doubles`, `ints`, etc.

Savitch limits his discussion to popular member functions of the STL `vector` class, such as `push_back`, `size`, `capacity`, `reserve` and `resize`. However there are other member functions such as:

`empty` which returns true if the `vector` is empty and false otherwise

`pop_back` which removes the last element of a `vector`

`back` which returns the last element of the `vector`.

Question 6

Discussion:

In this program we use C-strings to read in the user's first and last name. Both C-strings are then converted to lowercase before the full name is converted to pig latin.

C-strings use the '\0' character to indicate the end of a C-string. The C-string variables used to store the first and last names for example are therefore declared as

```
char first[21], last[21];
```

to allow 20 characters each for the first and last names as well as one position for the '\0'.

The program uses two functions, `convertToLowercase()` to convert a string to lowercase, and `pigLatin()` to convert a string to pig latin.

In function `convertToLowercase()` we use the null character '\0' as sentinel in the while loop that converts each character to its lower case.

Program listing:

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

// void convertToLowercase(char name[])
// Pre: name[] contains a C-string
// Post: name[] has been converted to lower case
void convertToLowercase(char name[])

//void pigLatin(char name[])
// Pre: name[] contains a C-string
// Post: name[] has been converted to pig latin
void pigLatin(char name[])

int main()
{
    char first[21], last[21], newName[41], copyFirst[21], copyLast[21];
    cout << "Please enter your first name: ";
    cin >> first;
    cout << "Please enter your last name: ";
    cin >> last;
    //make a copy of the first and last name for output purposes
    strcpy(copyFirst, first);
    strcpy(copyLast, last);
    //convert first and last name to lowercase
    convertToLowercase(first);
    convertToLowercase(last);
    //convert first and last name to pig latin
    pigLatin(first);
    pigLatin(last);
    //create new string with first and last name in pig latin
    strcpy(newName, first);
    strcat(newName, " "); //add space between first and last name
    strcat(newName, last);
```



```

    cout << "Dear " << copyFirst << " " << copyLast
    << " in pig latin your name is " << newName << endl;
    return 0;
}

void convertToLowerCase(char name[])
{
    int i = 0;
    while (name[i] != '\0')
    {
        name[i] = tolower(name[i]);
        i++;
    }
}

void pigLatin(char name[])
{
    char ch;
    if ((name[0] == 'a') || (name[0] == 'e') || (name[0] == 'i')
        || (name[0] == 'o') || (name[0] == 'u'))
    {
        name[0] = toupper(name[0]);
        strcat(name, "way");
    }
    else
    {
        ch = name[0];
        for (int i = 0; i <= strlen(name); i++)
            name[i] = name[i+1];
        name[strlen(name)] = ch;
        strcat(name, "ay");
        name[0] = toupper(name[0]);
    }
}

```

Input and corresponding output:

```

Please enter your first name: Erin
Please enter your last name: Jones
Dear Erin Jones in pig latin your name is Erinway Onesjay
Press any key to continue . . .

```

Question 7

7. (a) Discussion:

We adapted the sorting function to sort a vector from largest to smallest as follows:

The argument was changed from an array of ints to a vector of ints, as can be seen in the function headings:

```
void sort(vector<int>& v) and
```

```
int index_of_largest(const vector<int> v, int start_index)
```

Note that in order to return the sorted vector, it should be a reference parameter. Also note that since the size of a vector can be determined with the member function `size()`, the parameter `number_used` can be omitted from both functions.

We want to sort in descending order while the sorting function in Display 7.12 sorts in ascending order. Accordingly, both function and local variable names that refer to either 'smallest' or 'min' have been changed to 'largest' or 'max'. See for example function `index_of_largest` below:

```
int index_of_largest(const vector<int> v, int start_index)
{
    int max = v[start_index],
        index_of_max = start_index;
    for (int index = start_index + 1; index < v.size(); index++)
        if (v[index] > max)
        {
            max = v[index];
            index_of_max = index;
            //max is the largest of v[start_index] through v[index]
        }

    return index_of_max;
}
```

While these name changes aid in understanding the sorting order of the sorting function, it does not change the order in which the vector is sorted from ascending to descending. The crucial change to ensure that the sorting is done in descending order instead of ascending order, lies in changing the comparison

```
if (a[index] < min)
in function index_of_largest to
if (v[index] > max)
```

This change is highlighted in the code section above. The comments have also been adapted to reflect the changed sorting order.

7. (b) Discussion:

In this question you should have provided a facility to allow the user to specify whether input should be read from a file, or from the keyboard. Note that when input is read from the keyboard, we indicate the end of file character with CTRL Z, followed by pressing 'enter':

```
{
    //read input from the console
    cout << "Please enter list of values. Press 'enter' "
        << "after each value. Use CTRL Z to end." << endl;
    read_vector(list,cin);
}
```

In function `read_vector()` inheritance is used so that input can be done both from a file or from the keyboard. The formal parameter corresponding either to `cin` (console input) or `fin` (file input) therefore has to be of type `istream`:

```
void read_vector(vector<int>& v, istream& in_stream);
```

See Section 10.4 in Savitch for more detail.

The distinct elements in the vector are extracted by using a boolean function `found()` to determine whether or not a specific element in the original vector (`list`) occurs in the vector of distinct elements (`distinct`). Should a specific element not occur in the vector of distinct elements, it is added to `distinct`.

```
//extract distinct elements in list into vector distinct
vector<int> distinct;
for (unsigned int i = 0; i < list.size(); i++)
{
    if (!found(list[i], distinct))
        distinct.push_back(list[i]);
}
```

The vector of distinct elements is then sorted, and a third vector (`occurrences`) with the same number of elements as `distinct` is declared and initialised to 0.

```
//sort vector distinct
sort(distinct);

//declare a vector with distinct.size() elements and initialise
//each to 0
vector<int> occurrences (distinct.size());
```

A function `count()` is used to count the number of times each distinct element (stored in vector `distinct`), occurs in the original list of elements (vector `list`).

```
//count occurrences for each element in vector distinct
for (unsigned int i = 0; i < distinct.size(); i++)
    occurrences[i] = count(distinct[i], list);
```

Finally, the corresponding elements in vectors `distinct` and `occurrences` are output next to each other to show the number of times each distinct element occurs in the original list. We show output for input from the keyboard as well as for input from a file.

Program listing:

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <cstdlib>
#include <cstring>

using namespace std;

void read_vector(vector<int>& v, istream& in_stream);
//to read input values from a stream (file or console) into vector v
bool found(int x, vector<int> v);
//to determine if x occurs in vector v

int count(int x, vector<int> v);
//to count the number of times x occurs in vector v
```

```

void sort(vector<int>& v);
//Precondition: number_used <= declared size of the vector v.
//The vector elements v[0] through v[v.size() - 1] have values.
//Postcondition: The values of v[0] through v[v.size() - 1] have
//been rearranged so that v[0] >= v[1] >= ... >= v[v.size() - 1].

void swap_values(int& v1, int& v2);
//Interchanges the values of v1 and v2.

int index_of_largest(const vector<int> v, int start_index);
//Precondition: 0 <= start_index < v.size().
// Referenced vector elements have values.
//Returns the index i such that v[i] is the largest of the values
//v[start_index], v[start_index + 1], ..., v[v.size() - 1].

int main()
{
    vector<int> list;
    fstream fin;
    char answer;
    string filename;
    int next;
    cout << "Do you want to provide input via console or "
         << "using a file(c/f)?";
    cin >> answer;
    cout << endl;
    if (answer == 'f')    //read input from a file
    {
        cout << "Please enter filename: " << endl;
        cin >> filename;
        fin.open(filename.c_str());
        if (fin.fail())
        {
            cout << "Input file opening failed. \n";
            exit(1);
        }
        read_vector(list,fin);
        fin.close();
    }
    else    //read input from the console
    {
        cout << "Please enter list of values. Press 'enter' "
             << "after each value. Use CTRL Z to end." << endl;
        read_vector(list,cin);
    }

    //extract distinct elements in list into vector distinct
    vector<int> distinct;
    for (unsigned int i = 0; i < list.size(); i++)
    {
        if (!found(list[i], distinct))
            distinct.push_back(list[i]);
    }
}

```

```

//sort vector distinct
sort(distinct);

//declare a vector with distinct.size()elements and initialise
//each to 0
vector<int> occurrences (distinct.size());

//count occurrences for each element in vector distinct
for (unsigned int i = 0; i < distinct.size(); i++)
    occurrences[i] = count(distinct[i], list);

//output
cout << endl << 'N' << '\t' << "Count" << endl;
for (unsigned int i = 0; i < distinct.size(); i++)
    cout << distinct[i] << '\t' << occurrences[i] << endl;

return 0;
}

void read_vector(vector<int>& v, istream& in_stream)
{
    int next;
    while (in_stream >> next)
    {
        v.push_back(next);
    }
}

bool found(int x, vector<int> v)
{
    for (unsigned int i = 0; i < v.size(); i++)
    {
        if (x == v[i])
            return true;
    }
    return false;
}

int count(int x, vector<int> v)
{
    int counter = 0;
    for (unsigned int i = 0; i < v.size(); i++)
    {
        if (x == v[i])
            counter += 1;
    }
    return counter;
}

void sort(vector<int>& v)
{
    int index_of_next_largest;

```

```

        for (unsigned int index = 0; index < v.size() - 1; index++)
        {
            //Place the correct value in v[index]:
            index_of_next_largest =
                index_of_largest(v, index);
            swap_values(v[index], v[index_of_next_largest]);
        }
        //v[0] >= v[1] >= ... >= v[index] are the largest of the original
        //vector elements. The rest of the elements are in the remaining
        //positions.
    }
}

void swap_values(int& v1, int& v2)
{
    int temp;
    temp = v1;
    v1 = v2;
    v2 = temp;
}

int index_of_largest(const vector<int> v, int start_index)
{
    int max = v[start_index],
        index_of_max = start_index;
    for (int index = start_index + 1; index < v.size(); index++)
        if (v[index] > max)
        {
            max = v[index];
            index_of_max = index;
            //max is the largest of v[start_index] through v[index]
        }

    return index_of_max;
}

```

Output using console input:

Do you want to provide input via console or using a file(c/f)?c

Please enter list of values. Press 'enter' after each value. Use CTRL Z to end.

```

-12
3
-12
4
1
1
-12
1
-1
1
2
3
4
2

```

```
3
-12
^Z
```

```
N      Count
4      2
3      3
2      2
1      4
-1     1
-12    4
```

Press any key to continue . . .

Output using file input:

Do you want to provide input via console or using a file(c/f)?f

Please enter filename:
Q6.dat

```
N      Count
4      2
3      3
2      2
1      4
-1     1
-12    4
```

Press any key to continue . . .

Question 8

For this question you had to write a recursive function that returns the sum of the integers between any two integer numbers inclusive. This recursive function expects two integer parameters and returns a double.

Program Listing:

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

double sum (int m, int n)
{
    if (m == n)
        return m;
    else
        return (m + sum ( m + 1, n ) );
}

int main()
{
    cout << "The sum between 13 and 17 is:";
    cout << sum (13,17) <<endl;;
    cout << "The sum between 13 and 13 is:";
    cout<< sum(13,13) <<endl;
```

```

    cout << "The sum between 13 and 14 is:";
    cout<< sum(13,14) <<endl;

    return 0;
}

```

Output:

```

The sum between 13 and 17 is:75
The sum between 13 and 13 is:13
The sum between 13 and 14 is:27
Press any key to continue . . .

```

Discussion:

To solve this question, we need to determine the base case and the general case. The base case is when the solution can be obtained directly.

The base case:

If m is equal to n then we can immediately determine the sum to be m . For example the sum between 13 and 13 would be 13.

The general case on the other hand, is a little harder to deduce:

Consider the following example:

$$\text{sum}(13,17) = 13 + 14 + 15 + 16 + 17$$

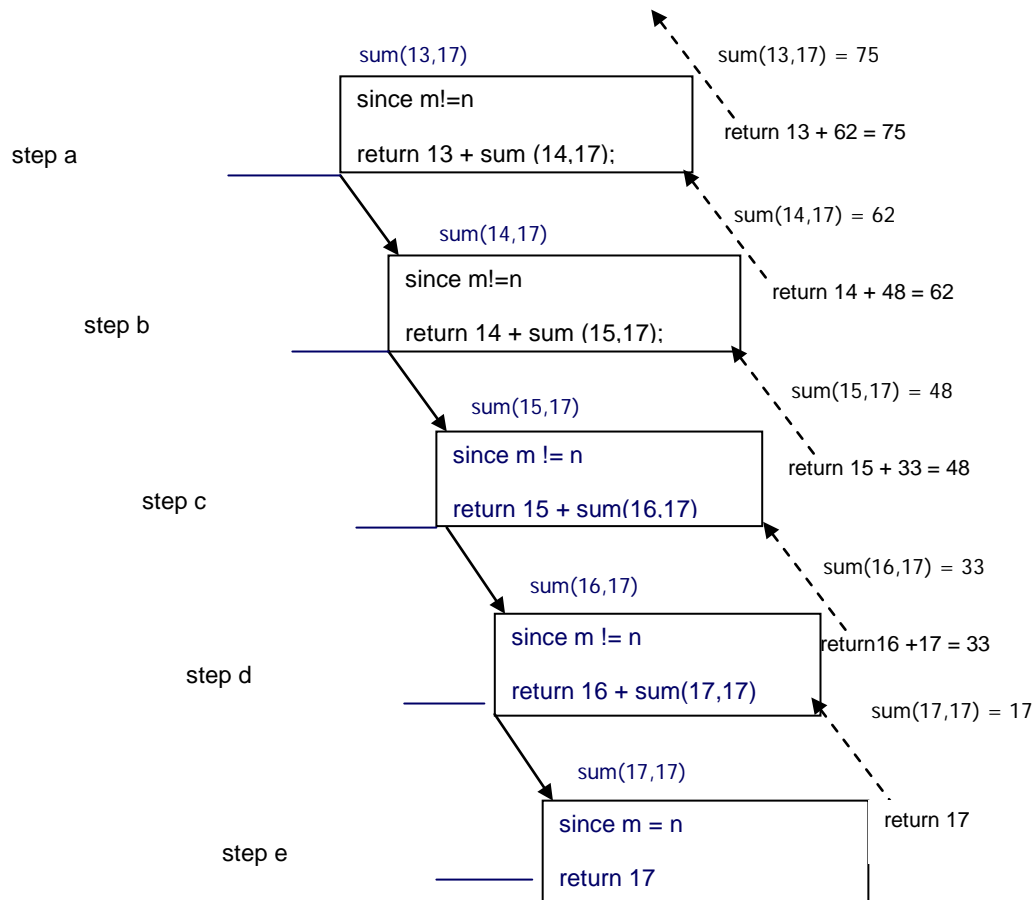
A more general formula can be written as:

$$\text{sum}(m,n) = m + (m+1) + (m+2) + \dots + (n-2) + (n-1) + n$$

Recursively:

$$\text{sum}(m,n) = m + \text{sum}(m+1,n)$$

The diagram below represents the recursive calls that will take place if 13 and 17 were passed as parameters to the `sum()` function. That is, we want to determine the sum between 13 and 17. The diagram shows that each `sum(m,n)` induces calls to `sum(m+1,n)`. For instance,



since $\text{sum}(13,17) = 13 + \text{sum}(14,17)$ - this induces calls to $\text{sum}(15,17)$, $\text{sum}(16,17)$ and finally $\text{sum}(17,17)$. With regard to the diagram below, recursion works first along the downward arrows until a given point is reached at which an answer is defined (the base case), and then works along the dashed upward arrows, returning values back to the calling function. For function `sum()`, the base case is when $m == n$.

So eventually the recursive calls will stop at $\text{sum}(17,17)$ - because an answer of 17 can be returned.

Hence $\text{sum}(16,17)$ is:

$\text{sum}(16,17) = 16 + 17 = 33$ hence $\text{sum}(16,17)$ returns 33

So $\text{sum}(15,17)$ is:

$\text{sum}(15,17) = 15 + \text{sum}(16,17) = 15 + 33 = 48$ hence $\text{sum}(15,17)$ returns 48

So $\text{sum}(14,17)$ is:

$\text{sum}(14,17) = 14 + \text{sum}(15,17) = 14 + 48 = 62$ hence $\text{sum}(14,17)$ returns 62

Therefore $\text{sum}(13,17)$ is:

$\text{sum}(13,17) = 13 + \text{sum}(14,17) = 13 + 62 = 75$ hence $\text{sum}(13,17)$ returns 75

More on Recursion:

Understanding recursion is difficult. Let us consider an everyday example. Suppose you were given a huge bag of coins and you had to determine how much money was in the bag.

As the bag is large, you prefer not to do the work by yourself. However, you have many willing friends. You divide the bag of coins into two heaps of coins and ask your friend "Could you please add up this one heap of coins? I've only given you half, so there's half the work to do". You, then give the other half to another friend, and say the same thing. Once both are done, they will give their answer to you, and you add their results.



Thus, you have broken down the problem into two smaller parts, and asked your friends to do the work.

Now those friends are smart too, so they divide their heap of coins into two parts (now each has two heaps of $\frac{1}{4}$ of the size) and each asks two of their friends to help. When their friends are done, they return their answer, and the result is summed. Now assume that each of their friends does the same and enlists more friends to help and this process goes on and on. Eventually, there is a heap of only two coins, and these are divided once again and given to two more friends, and those friends, seeing how silly the problem is now, just tell the first friend the only value on the coin. There's no need to ask any more friends, because you're down to one coin (this is the base case).

Thus, recursion is all about breaking a problem down, and solving that, and that smaller problem is solved by breaking it down some more, and trying to solve that. Eventually, you reach an easy solution (the base case), and return the solution.

©

Unisa 2017