UNIVERSITY EXAMINATIONS

UNIVERSITEITSEKSAMENS

UNISA university of south africa

COS1512
RCO1512

May/June 2016

INTRODUCTION TO PROGRAMMING II

Duration 2 Hours

75 Marks

EXAMINERS		
FIRST	MRS HW DU PLESSIS	MRS MA SCHOEMAN
SECOND	MS A THOMAS	

Closed book examination

This paper consists of 7 pages and 6 questions
Please make sure that you have all 7 pages with the 6 questions.

Instructions:

- Answer all the questions
- Do all rough work in the answer book
- The mark for each question is given in brackets next to the question
- Please answer the questions in the correct order If you want to do a question later, leave enough space
- Number your answers and label your rough work clearly
- Marks are awarded for part of an answer, so do whatever you are able to in each question

GOOD LUCK!

QUESTION 1 [5]

Do not copy the code in your answer book

Consider the following code fragment, and then answer the questions that follow:

```
1  char funnyName[10] =
          {'o', 's', 't', 'r', 'o', 'l', 'i', 't', 'c', 'h'};
2  char funnyWord[] = "geNONkiel@";
3  if (strcmp(funnyName, funnyWord))

4     funnyWord = funnyName;

5  else funnyWord[10] = 'z';
```

1.1 With the current declaration of `funnyName` in line 1, the instruction in line 3 does not give the desired result. How would you change the declaration of the variable `funnyName` so that the instruction in line 3 would be valid? (1)

1.2 Why is the instruction in line 4 invalid? Give the correct C++ instruction to copy the contents of `funnyName` to `funnyWord`. (2)

1.3 Under which condition will the corrected instruction in line 4 be executed? (1)

1.4 What is the problem with the C++ instruction in line 5? (1)

[5]

QUESTION 2 [4]

Function `sum()` is meant to be a recursive function to calculate the sum of the integers from 1 to a given integer, e.g. if the integer given is 3, it will calculate 3 + 2 + 1 = 6. Consider the program and then answer the questions that follow:

```
1   #include <iostream>
2.  using namespace std;
3   int sum(int num)
4.  {
5      if (num == 0)
6        return 0;
7      else
8      {
9          cout << "num " << num << endl;
10.
11     }
12  }
13. int main()
14  {
```

```
15.  int answer = 0;
16.  answer = sum(4);
17.  cout << "The answer is " << answer << endl;
18.  return 0;
19  }
```

2.1 Identify the base case in the recursive function `sum()` (1)

2.2 Write code for line 10 to implement the general case in the recursive function `sum()`. Write down the general case only - do not copy the program in your answer book. (2)

2.3 After completing `sum()` correctly (as stated in 2.2), what is the output generated when main() is executed? (1)

[4]

QUESTION 3 [7]

Consider the following code fragment which is assumed to be embedded in a complete and correct C++ program.

```
1     int *p1, *p2, x = 20, y = 10;
2     p1 = &x;
3     *p2 = x - 10;
4     *p1 = y + 5;
5     int *p3 = new int;
6     *p3 = *p1;
7     p2 = p3;
8     cout << *p1 << ' ' << *p2 << ' ' << *p3 << endl;
9     //release p2
```

3.1 The statement in line 3 is a risky statement. Why? (1)

3.2 What is the purpose of the statement in line 5? (1)

3.3 What is the purpose of the statement in line 2? (1)

3.4 What is the purpose of the * in line 1? (1)

3.5 What is the purpose of the * in line 4? (1)

3.6 Assuming that we do not execute line 3, what is the output after line 8? (1)

3.7 Write down a statement for line 9 to release `p2` to the freestore. (1)

[7]

QUESTION 4 [31]

Note Read through the whole of Question 4 below before you attempt to answer the questions that follow

Use separate compilation to define a class called `Champion` that represents a champion of an online game This class contains three member variables

- `name`, a string that holds the name of the champion
- `score`, an integer variable that holds the top score of the player
- `level`, an integer that holds the level reached in his best game

In addition, the class should contain the following member functions

- A default constructor that initializes `name` to an empty string, and score and `level` to `0`
- An overloaded constructor that accepts a champion's details and sets `name`, `score` and `level` to specified values
- A destructor that does not perform any action
- Accessor function `get_score()` to return the value stored in an object's `score` member variable
- An overloaded operator `>=` to compare a winner to the champion, to see if the winner could be the new champion The `>=` operator is implemented as a **`friend`** function with the following prototype

  ```
  bool operator>=(const Champion & champ1, const Champion & champ2)
  ```

  This function returns `true` if `champ1` has at least the same score and level reached as `champ2` and `false` if not

- An overloaded extraction operator `>>` (implemented as a **`friend`** function) so that it can be used to input values of type `Champion` from any file
- An overloaded insertion operator `<<` (implemented as a **`friend`** function) that displays a champion's name, game score and level reached

You should attempt the solutions as follows

4 1 Create the header file `champion h` that contains the `Champion` class specification (8)

4 2 Create the implementation of the class `Champion` including all the **`friend`** functions (11)

4 3 Demonstrate the class in an application program (`main()`) that is used to list all the winners that have equalled or bettered the current champion's score Allow the user to enter the score and level reached, of the current champion Use the overloaded

constructor to initialise the `Champion` object `current_champion` to the game score and level the user specified (initialize the name for this object to an empty string)

All the best players after a recent competition are stored in a file `Winners txt` Use a `while` loop to input the winners from `Winners txt`, use the overloaded operator >= to compare the scores and levels reached read from `Winners.txt` one by one with `current_champion`, and display a list of all the winners that have at least the same score and level of the specified champion's score and level (10)

4 4 Why is no code required for the destructor in this class? (1)

4 5 Why are the overloaded operators >=, >> and >> implemented as a **friend** functions? (1)

[31]

QUESTION 5 [14]

Consider the class specifications (interfaces) for the classes `SAtoUK` and `SAtoARICA` below Class `SAtoUK` represents flight details and requirements to visit the UK, and class `SAtoARICA` represents flight details and requirements to visit African countries

```
class SAtoUK
{
public
      SAtoUK (),
      SAtoUK (string name, string passport, string toFlightNr,
              string returnFlightNr, string toDate, string
              returnDate, string visa, int luggage);
      void set_details (string name, string passport),
      void set_flight (string toFlightNr, string returnFlightNr,
              string toDate, string returnDate);
      void set_visa (string visa),
      void set_luggage (int luggage);
      void get_flightDetails (string name, string passport, string
              toFlightNr, string returnFlightNr, string toDate,
              string returnDate) const;
      string get_visa ( ) const,
      int get_luggage ( ) const;
      void display_Info( ) const, //display name, passport number,
                     //flight numbers, to and return dates, visa info

private
      string Name,
      string Passport,
      string TODate,   //TO date
      string RETDate,  //return date
      string TONumber, //TO flight number
      string RETNumber //return flight number
      string Visa     //type of Visa required
      int KGS         //max weight for luggage
}

class SAtoAFRICA
{
public
      SAtoAFRICA (),
      SAtoAFRICA (string name, string passport, string toFlightNr,
              string returnFlightNr, string toDate, string
              returnDate, string yellowcard);
      void set_details (string name, string passport),
      void set_flight (string toFlightNr, string returnFlightNr,
              string toDate, string returnDate),
      void set_YF (string certificate);
                                      //yellow fever certificate details
      void get_flightDetails (string name, string passport, string
              toFlightNr, string returnFlightNr, string toDate,
              string returnDate) const,
      string get_YF ( ) const,
      void display_Info( ) const; //display name, passport number,
              //flight numbers, to and return dates, yellow fever
info
private
      string Name;
      string Passport,
      string TODate,
      string RETDate,  //return date
      string TONumber; //TO flight number
      string RETNumber; //return flight number
      string YF     //yellow fever certificate details
}
```

5 1 Create a base class `Travel` from which both classes `SAtoUK` and `SAtoAFRICA` can be derived Provide only the interface for the base class (6)

5 2 Code the interface for class `SAtoAFRICA` as derived from the class `Travel` Override the member function `display_Info()` for class `SAtoAFRICA` Do not provide any implementation (5)

5 3 Give an explanation of the difference between overloading a function and redefining a function Class `SAtoAFRICA` should override `display_Info( )` Is this function overloaded or redefined? Explain your answer (3)

[14]

QUESTION 6 [14]

Affordable Airlines uses a directory to keep track of their consumable parts that they use to repair aircrafts They use this directory to order parts ahead of time, so that they don't run out of stock at any given time The part name is represented by a code and the number of the specific part still available in stock is represented by a number, for example

Part code	Number available
416000	10
416001	26

416002	33
416003	17

The following class interface presents an approach to implementing the above scenario

```
    class Directory {
       public
              Directory();
              void AddPart(int code, const int &number),
              int Check (int code)  const,
       private
              vector<int>   Code,
              vector<int>   Number,
    };
```

The class `Directory` has the following operations (member functions)

- `AddPart()` - adds a code and number for new parts to the directory
- `Check()` - retrieves the number of parts available in store for a specified code, for example `Check(416002)` would return `33`

6 1 Provide an interface of the template class `Directory<TCode,TNumber>` In other words, re-design the `Directory` interface so that it may be used to create a `Directory` containing codes and numbers of different types For example they may want to change the part code to an alphanumeric value or a variable of type `double` and `Number` may have to change to `double` Note that the code and the number of items in stock may most likely be of different types hence we need two different template arguments (5)

6 2 Implement the `Check()` function of the template class `Directory <TCode,TNumber>` Use the `assert()` function to validate that the number of items in stock obtained from the directory, is greater than `0` (7)

6 3 Provide a declaration for a `Directory` that has codes of type `string` and availability numbers of type `double` (2)

[14]