UNIVERSITY OF JOHANNESBURG
FACULTY OF SCIENCE

COMPUTER SCIENCE	**APK CAMPUS**
CSC01B1 **INTRODUCTION TO DATA STRUCTURES (C++)** **SEMESTER TEST 2** **AFTERNOON SESSION**	
8 OCTOBER 2021	

EXAMINER	**Prof. DA Coulter**
MODERATOR	**Prof. DT van der Haar**

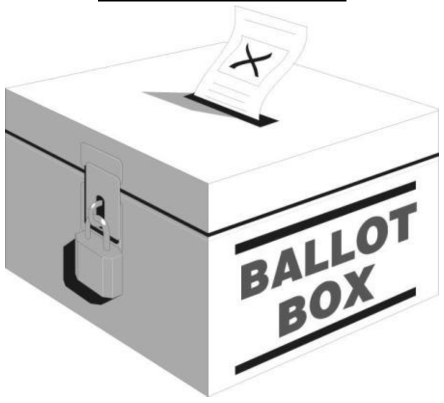
TIME	**3 Hours**	**MARKS**	**100**

Please read the following instructions carefully

1. **You must complete this test yourself within the prescribed time limits.**
2. **You are bound by all university regulations please special note of those regarding assessment, plagiarism, and ethical conduct.**
3. **While this is an open book test you may not directly use any code from any source including your own previous submissions. All code must be written by yourself during this the test.**
4. **You must regularly save your work to the VPL system https://acsse.cloud during the assessment. We recommend working directly on VPL in case the computer you are using crashes so that your work is still accessible. You may access your VPL login details under marks. Remember to leave out the "/0" at the end.**
5. **You must complete and submit the** `"Honesty Declaration : Online Assessment"` **document along with your submission to EVE. No submissions without an accompanying declaration will be marked.**
6. **Your completed code at the end of the assessment must be submitted in a zip archive to EVE. This zip file is the one downloaded from the VPL system. You do not need to rename the submission. When you have uploaded to the zip file to EVE your test is over. You may only upload the zip file once.**
7. **No communication concerning this test is permissible during the assessment session except with Academy staff members.**
8. **This paper consists of 4 pages excluding the cover page**

Note: Although there are 120 marks available the maximum mark which can be achieved is 100

Poll Position



The Utopian Electoral Commission is preparing for the upcoming election. Utopia has a two-party electoral system with citizens only being allowed to vote for either the Utopian Authoritarian Party (UAP) or the Authoritarian Party of Utopia (APU). The commission is interested in conducting various analyses about voting districts to identify potential problem areas and also to allocate staff and resources.

Due to the recent introduction of anti-gerrymandering laws all voting districts are uniform squares and have been projected onto a two-dimensional grid. Each Voting District within the grid contains the following information.

Field	Details
District ID (row, column)	Pair of integers each greater than or equal to 0. Must be unique i.e. no two districts can have the same combination of District ID components as another.
Population	Integer value greater than or equal to 0
UAP support percentage	Real value in the range [0, 100]
APU support percentage	Real value in the range [0, 100]
Undecided support percentage	Real value in the range [0, 100]
NOTE: The three support fields must add up to 100%	

- You must create a system which will generate a two-dimensional data structure which stores information related to Voting Districts in a Voting Grid as described above.
- The Voting Grid can be analysed by Voting Analysers which each perform a different analysis as described below
    - A battleground analysis displays different symbols for each Voting District depending on how close the support for the two parties is.
        - If the difference is less than 5% the character '!' is output
        - If the difference is from 6% to 10% the character '%' is output
        - If the difference is from 11% to 20% the character '*' is output
        - If the difference greater than 21% the '.' character is output

Note: Although there are 120 marks available the maximum mark which can be achieved is 100

- A potential fraud analysis displays different symbols for each district as follows:
  - If the sum of supporters for each party and the undecided voters does not add up to approximately 100% (within 1%) then the Voting District's ID is output (this analysis takes the form of a list not a grid)
- A predicted outcome analysis displays the following for each cell in a grid format:
  - If UAP has the highest support in a district it will be displayed using a 'U' character.
  - If APU has the highest support in a district it will be displayed as an 'A" character.
  - If the undecided voters have the highest support in a district, it will be displayed as a '?' character.
  - After displaying the grid the total number of voters for the two parties is displayed (does not need to be rounded off).

You have been provided with the following in order to assist you in developing the system to the UJ standards. Note the VPL system will make the file automatically available to you when you click `run`:

- An `input.txt` file contains the information for the system in the following format where EOL represents the end of line marker:
  - `ROWS COLS EOL`
  - `UAP-SUPPORT APU-SUPPORT POPULATION … EOL`
    - This information is given for each district on a row-by-row basis
    - You will need to generate the District ID (row, column) values yourself on the basis of their position in the file.
    - The undecided value must be calculated based on the UAP-SUPPORT and APU-SUPPORT values (remember the support values must total to 100%).
- `electionMonitor.dll` which contains the following C function with name mangling disabled: `bool isFreeAndFair();`

In order to meet the requirements of this task the project manager has broken the overall problem into several sub tasks (see accompanying mark sheet for more details):

- T0 – Design: You must create a UML 2.0 Class Diagram according to the requirements of the other tasks. It must be saved in a PDF file.
- T1 – Class Basics: You must create a class to represent the `VotingGrid`. The class must make use of the principles of good design known to you. It must additionally make use of a dynamic two-dimensional array of structures in which to store the data.
- T2 – The `VotingGrid` class has a collection of one of each kind of `VotingAnalyser`s which implement the `analyse` method that will return a string representation of the grid. The `VotingGrid` delegates its own `analyse` method to the contained `VotingAnalyser`s. The `VotingGrid` class must manage the life cycle of the `VotingAnalyser` objects.
- T3 – Modify the previously created classes to use an inheritance hierarchy with polymorphism as follows:
  - A `VotingAnalyser` is an abstract base class with a string returning pure virtual function called `analyse`

Note: Although there are 120 marks available the maximum mark which can be achieved is 100

  - A `BattleGroundAnalyser` is a kind of `VotingAnalyser` that overrides the `analyse` method to perform the battleground analysis described previously.
  - A `PotentialFraudAnalyser` is a kind of `VotingAnalyser` that overrides the `analyse` method to perform the potential fraud analysis described previously.
  - A `PredictedOutcomeAnalyser` is a kind of `VotingAnalyser` that overrides the `analyse` method to perform the predicted outcome analysis described previously.
  - The `VotingGrid` 's `analyse` method now must polymorphically call each of the contained `VotingAnalyser analyse` functions and return their combined output.
- T4 – Add the following to your `VotingGrid` class
  - An overloaded assignment operator which performs a deep copy of the underlying array.
  - Support for using the `!` operator to reset the grid:
    - The size of the grid stays the same
    - The district ids (row, column) stay the same
    - The population is set 0
    - The support percentages for the two parties are set to 0
    - The undecided value is set to 100
  - Support for double indexing by way of an overloaded function invocation operator "o(row, column)".
- T5
  - Define a `VotingDistrict` structure as per the given table with alignment padding disabled.
  - Add a member function to the `VotingGrid` class which will load the information from the provided text file.
  - Add a member function to `VotingGrid` which will save the contents of the two-dimensional array to a binary file of records.
  - In all cases the functions will take the name of the file as a parameter and close the file when it has finished.
- Additionally you have been provided with a `main.cpp` file which demonstrates the functionality of the class (you may edit this file as needed).
- Bonus
  - See descriptions on following page
  - Only one bonus may be attempted
  - The bonus must be submitted as a separate submission on EVE
  - The VPL system is Unix based so the DLL bonus question cannot be completed on the VPL.

Note: Although there are 120 marks available the maximum mark which can be achieved is 100

- Identified classes (2)
- Attributes (2)
- Operations (2)
- Has-a relationships (2)
- Is-a relationships (2)

T0 - UML Class Diagram [10]

- Constructors (5)
- Destructors (5)
- Information hiding (5)
- Use of .h and .cpp files (3)
- Comments (2)

T1 - Class Basics [20]

- Create contained class
- Has-a relationship (5)
- Lifecycle management (5)
- Delegation (5)
- Algorithm (5)

T2 - Composition & Delegation [20]

- Abstract base class (5)
- Overriding (5)
- Polymorphism (5)
- Member visibility (5)

T3 - Inheritance & Polymorphism [20]

- Assignment = (5)
- Negation / Reset ! (5)
- Double Indexing ()(5)

T4 - Operator Overloading [15]

- Define structure (5)
- Write binary file (5)
- Read text file (5)

T5 - File Handling [15]

You may only select <u>one</u> of the following bonus sections:

Bonus - Templates [20]

- Make the entire VotingDistrict class templated and every member function must be implemented as a templated function.

Bonus - DLLs [20]

- Provide code for an executable which loads and runs the file from the given DLL using explicit linking (10) and implicit linking (5).
- Batch file (5) - The executables must both be compiled from a .BAT file

Bonus - Exception Handling [20]

- Use exception handling to handle errors involving impossible values in the analyses using an inheritance hierarchy of exception classes. Update main accordingly.

Note: Although there are 120 marks available the maximum mark which can be achieved is 100