SKILLS CANADA COMPETITION
2020 Provincial Skills Competition

Coding

Instructions and Information

This Contest will test your knowledge in the following areas:

Problem Solving/Knowledge • Flowcharts/Pseudocode • User Requirements • Testing • Presentation of Software Developed • Math Programming Skills • Variables • File I/O • Arrays • Control Structures • Console Based Applications

Please read this section carefully.

- You can use Visual Studio 2019 community(C#,C++ or C), Java 1.8 or Python 3.6 interpreter.
- You only can use all the build-in functions/classes which come from the standard libraries of the corresponding SDKs:
  - **Only Standard Python Library(https://docs.python.org/3/library/) is allowed**
  - **Only Standard Java SE 8 APIs(https://docs.oracle.com/javase/8/docs/api/) are allowed**
  - **Only the APIs/Classes in .NET 4.7 are allowed**
  - **Only Microsoft implementation of the C runtime library (CRT) and Microsoft implementation of the C++ standard library shipped with Visual Studio 2019 are allowed**
- There should be no communication between candidates.
- The facilitator is not expected to answer any questions
- You will make sure all the Textbooks, Documents and Notes are available to you before the contest. **Please make sure you have the offline version of SDK document in your computer before the contest start**.
- Everything that you want marked will be compressed in to a zip file and submitted via an OneDrive file request link.
- Before the contest starts, you will receive an email from the facilitator which includes how to get the contest package
- Save the file you create with the names provided.
- If you get stuck on a question, move on as time will be short.

Time Lines

8 :15am – 8:30am	Deliver/Receive the contest package
8:30am – 8:45am	Orientation
8:45am – 12:00pm	Contest
12:00pm – 12:30pm	Lunch
12:30pm – 3:30pm	Contest

Coding Question and Deployment

You will be creating a **Terminal or GUI** Based application for helping Saskatchewan Skills manage their competitions. This application can be written in either C#, Python, C, C++ or Java and submitted as either an .exe, .py or .jar file. Hand in the **source code (with project files if available)**, documentation and deployment files or installations script.

The overall goal is creating a **simple ISAM database system**.

ISAM (an acronym for **indexed sequential access method**) is a method for creating, maintaining, and manipulating computer files of data so that records can be retrieved sequentially or randomly by one or more keys. Indexes of key fields are maintained to achieve fast retrieval of required file records in Indexed files. IBM originally developed ISAM for mainframe computers, but implementations are available for most computer systems.

The first step in managing the competition is creating a (set of) Binary or Text file (s) to store contestant data. Your program will be use this file data for basic CRUD functions (create, read, update, delete).

- You can design your system to
  - Store all the data in one file
  - Or split it between two files (Split the "Competition" field to a separate file)
  - Or even have multiple files(Implement the Indexes file to have better random access performance)

  Please see details in the "Additions" section. There will be extra marks for the multiple files implementation.

- You can choose between the binary File IO and Text File IO. There will be **extra marks for the binary file IO**.

The file must include the following:

- ID: A generated incrementing 0 padded 8 digit number. For example, "123" should be padded as "00000123". "12345" should be padded as "00012345".
- First Name: The contestant's first name, mandatory.
- Last Name: The contestant's last name, mandatory
- Email Address: The contestants email address.
- School District: The list of available school districts should be read in from a file containing the following options (Chinook, Creighton, Good Spirit, Horizon, Ile a la Crosse, Living Sky, Lloydminster, North East, Northern Lights, Northwest, Prairie South, Prairie Spirit, Prairie Valley, Regina, Saskatchewan Rivers, Saskatoon, South East, Sun West)
- Birthday: Date of the contestant's birth. Format dd/mm/yyyy.
- Competition : What event the contestant is competing in. (example: Coding, IT Software, Networking)
- Score: The score the contestant received (assume it is a percent value).

The application itself can function in one of two ways: either by **displaying a user interface**, or by **accepting command line arguments**.

It has to have the following functionality.

- Add Contestant.
    - The application should not allow the same contestant to be added twice.
    - The School District should be valid
- Remove Contestant by ID.
- Search for Contestant by Last Name, Age, and School District.
- Display Contestants Sorted by Last Name, First Name.
- Display the top 3 Contestants for any given competition.
- Usage instructions or a help option.

Deployment (No Presentation)

- Create a README.md file that contains information on how to use your application.
- System design document(Flowcharts)
- Your own flavor with a note explaining your optional Additions.

When submitting the contest, put ALL files into ONE Zip file. This includes the files you used for testing the application.

Additions

Above are the mandatory requirements for the application. The following is a list of optional additions.

- Split the “Competition” field to a separate file. Allow us to list all competitions and display the relevant information.
- Implement the Indexes file to have better random access performance.
- CSV/TSV based data import/export. The export CSV file should be readable by Excel.
- Full text search. For example, “South” should hit all Contestants whose name contains “South” or belong the “Prairie South” or “South East” District.

Coding Rubric (60%)		Mark	Mark Received
1.	Coding Standards / Code Quality	2	
2.	ID is a 0 padded number	2	
3.	Validations: id(8), Name not empty, numeric amount, email is valid	6	
4.	Required Fields	2	
5.	Search by Age	4	
6.	Search by School District	3	

7.	Searching by last name	3	
8.	Help – clean and useful	5	
9.	Adding new Contestants	5	
10.	Remove by Name	3	
11.	Remove by ID	3	
12.	School District Restrictions	2	
13.	Display Contestants	2	
14.	Display Contestants by Last Name, First Name, School District and Age	8	
15.	Display Top 3 Contestants	4	
16.	Random File IO – Read, Seek, Write (Get 3 if only using Binary File IO)	3	
	Total	**57**	
Additions			
18.	Implement the Indexes file to have better random access performance	3	
19.	Split the "Competition" field to a separate file (Mater-Detail, only with multiple file implementation).	2	
20.	CSV/TSV data import/export.	2	
21	Full text search	2	
	Total	**9**	
Deployment			
	Deployment (No Presentation/ Judge run demo)		
22	- README File	2	
23	- User Interface - Ease to use	2	
24	- System Design Documents (flowcharts)	6	
	Total	**10**	

	Code Efficiency		
25	Implementation of Index(Memory and/or File)	**5**	
26	Implementation of advanced Index (Tree, Hash, etc.)	**5**	
27	In memory cache	**4**	
28	Implementation of multiple Indexes(Memory and/or File)	4	
29	Performance testing/measurement script which demonstrates the efficiency of the implementation. Please submit testing data (you should generate a lot of testing records), Performance testing script, and performance testing result.	6	
	Total	**25**	
	Final Mark	**100**	