LaSalle College
Montréal

PROJECT: 30%

Course Identification

Name of programs – Codes:	**COMPUTER SCIENCE TECHNOLOGY** **PROGAMMING (420.BP)** **INFORMATION TECHNOLOGY** **PROGRAMMER ANALYST (LEA.3Q)**
Course title:	**ALGORITHMS AND PROGRAMMING**
Course number:	**420-AP1-AS**
Group:	**07214**
Teacher's name:	**Asma Aouichat**
Semester:	**Autumn 2024**

Standard of the Evaluated Competencies

Statement of the evaluated competency – Code
The achievement context of the competency *Use programming languages* - 00Q2:

- For problems that are easily solved
- Using basic algorithms
- Using a debugger and a functional test plan

The achievement context of the competency *Interact in a professional setting* - 00SE:

- In various types of work settings
- Using laws, code of ethics and corporate policies.

1- Project Aim:

The aim of this project is to enhance your C# programming and algorithmic problem-solving skills by developing a Factoid Question Answering (Q&A) system. You will apply techniques such as question analysis, sentence similarity computation, and answer extraction to build a system that answers fact-based questions from a given text. The project will challenge your understanding of algorithms and rule-based systems, allowing you to implement solutions without relying on advanced techniques in artificial intelligence.

Additionally, you are encouraged to incorporate basic visualizations that demonstrate how the system processes a question step-by-step, including how it retrieves and extracts answers from the text. Enhancing the visualizations for better user interaction may result in bonus points.

The final implementation should be a C# console application, and your work must be submitted through the Leo platform along with a detailed report describing your analysis of the project, adhering to the provided submission instructions.

2- General Notes:

You are required to submit a complete report that provides detailed explanations for each aspect of the project, focusing on the solutions and design decisions made to ensure the effectiveness and functionality of your implementation. Rather than describing the code, a good report should explain the rationale behind the approaches used, the challenges encountered, and the problem being solved during the tests. You are encouraged to include comparative analysis where applicable, showcasing the performance of different techniques employed in the system. Additionally, the project code must be well-commented and submitted electronically as a single package, following the specific submission guidelines provided.

The project will be completed by teams of 3 students, with one member designated as the Team Leader. The Team Leader will handle all communications with the course instructors. The report should also include a table that clearly outlines each team member's contributions. Keep in mind that all team members must participate in all aspects of the project, and it is unacceptable for a student to only focus on report writing. Active involvement in coding, testing, and designing is mandatory for everyone.

Each team will give a demo of the project at College LaSalle, where individual grades will be assigned based on each student's contribution and their ability to answer questions during the demo.

3- Practical Aspects:

As outlined above, the project work must be done in teams of exactly 3 students. Any other grouping will not be accepted and will result in penalties. The following steps must be followed:

- **Team Formation:** The team leader must send an email to me no later than Monday, **20/10/2024** at 11:59 p.m. In this email, the team leader should list the **name, first name and email address** of each team member. Ensure that your team has exactly 3 members, as having fewer or more will negatively affect your project grade.
- **Submission of Work:** Both the **report** and the **code** must be submitted on **Leo** as a **single compressed folder**. The folder name should be the **team leader's name**. It is the team's responsibility to ensure that the submission is complete and includes both files (PDF and code). Missing files or late submissions will be penalized, and if the submission is empty, a grade of **0** will be given for the relevant parts of the project.

Make sure to follow these instructions carefully to avoid any unnecessary penalties.

4- Submission Guidelines:

You are required to submit ONE folder through Leo, which should contain both your C# console application code and the report (PDF file; please check the report's structure).

5- Project Submission Deadline:

- Your project must be submitted by **Thursday , 21 November 2024, no later than 11:59 p.m.** via **Leo**.
- For every **24 hours late**, **25%** of your grade will be deducted.
- No project will be accepted after **Saturday, 23 November 2024, at 11:59 p.m.**

6- Grading of the mini-project :

Competency:	Use programming languages - 00Q2	
Elements of the competency		
420.BP	***Performance criteria specific to each element***	
1. Analyze the problem.		**/30**
1.1 Correct breakdown of the problem		/10
1.2 Proper identification of input and output data and of the nature of the processes		/10
1.3 Appropriate choice and adaptation of the algorithm		/10
2. Translate the algorithm into a programming language.		**/25**
2.1 Appropriate choice of instructions and types of elementary data		/5
2.2 Efficient modularization of code		/5
2.3 Logical organization of instructions		/5
2.4 Compliance with the language syntax		/5
2.5 Computer code consistent with the algorithm		/5
3. Debug the code.		**/5**
3.1 Efficient use of the debugger		/1
3.2 Identification of all errors		/1
3.3 Astute choice of debugging strategies		/1
3.4 Relevance of the corrective actions		/1
3.5 Clear record of solutions to the problems encountered		/1

Competency:	Interact in a professional setting - 00SE	
Elements of the competency 420.BP	***Performance criteria specific to each element***	
1. Establish professional relationships with users and clients.		**/20**
1.1 Attitudes and behaviours that demonstrate the ability to listen		/5
1.2 Adaptation of the level of language to the situation		/5
1.3 Observance of rules of politeness and common courtesy		/5
1.4 Observance of the client-based approach		/5
2. Work within a multidisciplinary team.		**/10**
2.1 Attitudes and behaviours that demonstrate respect, openness and a collaborative spirit		/1
2.2 Effective communication with all team members		/1
2.3 Proper performance of assigned tasks		/1
2.4 Compliance with rules for optimal team function		/1
2.6 Compliance with application programming and network management standards, methods and best practices		/1
2.7 Observance of the limits of the scope of professional intervention and respect for the expertise of team members in other occupations		/4
2.8 Adherence to deadlines		/1
3. Become familiar with the legal obligations and rules of professional ethics.		**/10**
3.4 Determination of the measures appropriate to the situation		/5
3.5 Compliance with laws, codes of ethics and corporate policies		/5

SubTotal out of 100 (before deduction due to the errors)	**/100**
Deduction due to the spelling and grammar mistakes (0.5 for each mistake / maximum 5%)	**/100**
Total of the evaluation out of 100	**/100**
Total of the evaluation out of 30	**/30**

Description of the project:

Question Answering System Project

1. Description

This project aims to build a simple Factoid Question-Answering (Q&A) system using rule-based methods. A Q&A system enables computers to read, understand, and interpret text to answer specific questions. In particular, a Factoid Q&A system handles questions seeking factual information, such as names, dates, locations, or quantities.

The system works with two inputs: a text and a question. It starts by analyzing the question (using the **QuestionAnalysis** module) to identify the answer type (e.g., a person's name) and filters out irrelevant words, such as stop words. Then, the **InfoRetrieving** module breaks down the input text into sentences and selects the one most relevant to the keywords in the question. This sentence is then passed to the **AnswerExtraction** module, which extracts the entity that matches the expected answer type. This entity is returned as the final answer to the question.

2. Basic Concepts

Text: A text is a collection of sentences, each ending with a period ('.'). For example: "The history of programming languages spans from early mechanical computers to modern software tools."

Question: A question is an interrogative expression ending w. ith a question mark ('?'). For example: "When was Konrad Zuse born?"

Factoid Question: A factoid question seeks specific factual details, often starting with:

- "Who" (for a person's name)
- "When" (for a time or date)
- "Where" (for a location)
- "How many" or "How much" (for numbers)

Examples include:

- "Who created the first high-level programming language?"
- "When was Konrad Zuse born?"

Segment: In this context, a segment refers to a part of a text, such as a sentence or a word.

Answer: The response to a factoid question, which is typically a specific piece of information (e.g., a name, location, or date) extracted from the text.

Answer Type: This refers to the category of information expected in the answer (e.g., person's name, location, number, date, or time).

Sentence Similarity: This is a measure of how closely two sentences resemble each other. In this project, we focus on *lexical similarity*, which measures the overlap of words between two sentences. High similarity means the sentences share many identical or similar words. Techniques like Jaccard similarity, Cosine similarity, or TF-IDF can be used for this purpose. For instance:

- Sentence 1: "The cat sat on the mat."
- Sentence 2: "The cat rested on the mat." These sentences share most words, so they have high lexical similarity.

Stop Words: These are common words (e.g., "the," "and," "is") that are usually removed during text analysis as they add little meaning.

Named Entity Recognition (NER): This NLP task identifies and classifies specific entities in text, such as people, organizations, locations, dates, and numbers. For example:

- "Albert Einstein" (Person)
- "New York" (Location)
- "January 1, 2020" (Date)
- "$1,000" (Monetary value)

3. Key Components of the Question-Answering System

We would establish the simple version of a Facroid Q&A system. To do so, it is essential to delineate the following fundamental components:

3.1. Question Analysis

Factoid questions generally seek specific factual data. They often begin with keywords like:

- "Who" (for a person's name)
- "When" (for a date or time)
- "Where" (for a location)
- "How many/How much" (for numeric details)

To identify the type of answer expected, we implement a function called determineFactoidType that analyzes the question's structure and returns the type of entity sought. For example, a question starting with "Who" will return "Name of person."

Tasks:

1. Provide an analysis of the question analysis module.
2. Write an algorithm (pseudo-code) that generates the expected answer type (e.g., person's name, date, location).
3. Write a C# function that takes a question as input and returns the expected answer type.

3.2. Sentence Similarity Module

This module retrieves the sentence from the text that best matches the question. The text is split into sentences, and each sentence is compared with the question based on common keywords, ignoring stop words. The module then selects the sentence with the highest lexical similarity to the question. To achieve this, we implement the following:

- **removeStopWords**: A function that removes common stop words from a given text.
- **calculateSimilarity**: A function that calculates the similarity between two sentences by counting the number of common words and dividing this by the minimum word count of either sentence.

Tasks:

1. Write a pseudo-code algorithm for removing stop words.
2. Write a C# function for removing stop words.
3. Provide an analysis of how to calculate sentence similarity.
4. Write a pseudo-code algorithm for calculating sentence similarity.
5. Write a C# function for calculating sentence similarity.
6. Write a C# function that finds the most similar sentence from a text.
7. Test the Sentence Similarity Module with various inputs.

3.3. Answer Extraction Module

The **AnswerExtraction** module is responsible for extracting the correct entity from the most relevant sentence. In a factoid Q&A system, this answer is typically a person's name, location, number, date, or time. We use a rule-based approach to extract these entities instead of relying on advanced Natural Language Processing libraries.

For example:

- **Person**: Terms that start with an uppercase letter followed by lowercase letters (e.g., "Steve Jobs").
    - The function getPersonName scans for such terms and returns the person's name.
- **Location**: Words in all uppercase letters (e.g., "CUPERTINO").
    - The function getLocation searches for these uppercase location names.

- **Date/Time**: Specific date or time formats (e.g., "YYYY-MM-DD").
    - The function getDateTime identifies valid dates or times.
- **Number**: Includes monetary values, percentages, or other numeric values.
    - The function getNumber detects these and returns the relevant number.

Example Usage: Consider the text:

> "*Apple Inc. was founded by Steve Jobs and Steve Wozniak in CUPERTINO, CALIFORNIA, on 1976-04-01. The company initially raised $1,000 to develop their first product. In 2023, Apple reported a 15% revenue increase, reaching a total of $387.53 billion."*

→ **getPersonName** will identify "Steve Jobs" and "Steve Wozniak."

→ **getLocation** will find "CUPERTINO."

→ **getDateTime** will extract "1976-04-01" and "2023."

→**getNumber** will detect "$1,000", "15%", and "$387.53."

By applying these rules, the system can provide accurate answers to factoid questions, ensuring precise and reliable responses.

Tasks:

1. **Provide an analysis of the answer extraction module.**
2. **Person's Name Extraction**: Implement extraction of person names based on the rule of uppercase words followed by lowercase letters.
    - **2.1. Write a pseudo-code algorithm:** Create pseudo-code that scans a sentence and identifies person names by looking for terms that begin with a capital letter followed by lowercase letters.

2.2. **Write a C# function:** Implement the getPersonName function in C# using the rule defined in the pseudo-code. The function should take a sentence as input and return any identified person names.

3. **Location Extraction**: Extract locations based on uppercase words in the sentence.
   3.1. **Write a pseudo-code algorithm:** Create pseudo-code to identify words in all uppercase (e.g., "CUPERTINO") and extract them as locations.
   3.2. **Write a C# function:** Implement the getLocation function in C# that scans a sentence and returns the locations in uppercase letters.
4. **Date/Time Extraction**: Extract specific date or time formats (e.g., "YYYY-MM-DD") from sentences.
   4.1. **Write a pseudo-code algorithm:** Create pseudo-code for identifying dates and times using specific formats like "YYYY-MM-DD" or common time expressions.
   4.2. **Write a C# function:** Implement the getDateTime function in C# to extract valid date/time formats from a sentence. The function should scan the text and return any identified dates or times.
5. **Number Extraction**: Extract numeric values, including monetary amounts, percentages, and other numbers.
   5.1. **Write a pseudo-code algorithm:** Create pseudo-code to identify numbers (e.g., "$1,000", "15%") and return them as numeric entities.
   5.2. **Write a C# function:** Implement the getNumber function in C# that detects and returns monetary values, percentages, or other numeric entities from a sentence.
6. **Testing the Answer Extraction Module**: Perform testing of the entire module to ensure that each entity (person name, location, date, number) is correctly extracted.
   - Prepare a variety of sentences with names, locations, dates, times, and numbers to validate the correct extraction of entities.
   - Test edge cases, such as names with multiple capitalized words, locations with mixed-case letters, and unusual date formats.
   - Analyze the performance of the rule-based system compared to real-world text to ensure robustness and accuracy.

7. **Write a C# function that applies the rules for all entity types:** Create a function that combines all the entity extraction functions (getPersonName, getLocation, getDateTime, getNumber) and applies them to the most relevant sentence to extract the correct answer.

4. Test Your Q&A system:

In this project, you are tasked with developing a simple C# console application to test the functionality of your Question-Answering (Q&A) system. The system should allow users to ask factoid questions based on the following text:

Text: *"The history of programming languages spans from documentation of early mechanical computers to modern tools for software development. Early programming languages were highly specialized, relying on mathematical notation and similarly obscure syntax. Throughout the 20th century, research in compiler theory led to the creation of high-level programming languages, which use a more accessible syntax to communicate instructions. The first high-level programming language was created by Konrad Zuse in 1943. The first high-level language to have an associated compiler was created by Corrado Böhm in 1951. Konrad Zuse was born on 1910/06/22, in GERMANY, and was a notable civil engineer, pioneering computer scientist, inventor, and businessman."*

Tasks:

1. **Your program should allow the user to ask multiple factoid questions**, such as:
   - *When was Konrad Zuse born?*
   - *Where was Konrad Zuse born?*
   - *Who is the creator of the first high-level programming language?*
2. **Report Enrichment**:
   - **Discuss three limitations** of your system and potential solutions.
   - **Identify and elaborate on the module** that posed the most challenges during this project.