NYU, Tandon School of Engineering
CS Bridge — Winter 2024

Exam 1

Thursday, February 8, 2024

- This exam has 14 questions, with 100 points total.

- You have **two hours**.

- **You should submit your answers on the Gradescope platform** (not on NYU Brightspace).

- **It is your responsibility to take the time for the exam** (You may use a physical timer, or an online timer: https://vclock.com/set-timer-for-2-hours/).
  Make sure to upload the files with your answers to gradescope BEFORE the time is up, while still being monitored by ProctorU.
  We will not accept any late submissions.

- In total, you should upload 3 '.cpp' files:
  - One '.cpp' file for questions 1-12.
    Write your answer as one long comment (/* … */).
    Name this file 'YourNetID_q1to12.cpp'.
  - One '.cpp' file for question 13, containing your code.
    Name this file 'YourNetID_q13.cpp'.
  - One '.cpp' file for question 14, containing your code.
    Name this file 'YourNetID_q14.cpp'.

- **Write your name, and netID at the head of each file.**

- This is a closed-book exam. However, you are allowed to use:
  - Visual Studio Code (VSCode) or Visual-Studio or Xcode or CLion. You should create a new project and work ONLY in it.
  - Two sheets of scratch paper.

  Besides that, no additional resources (of any form) are allowed.

- **You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.**

- Read every question completely before answering it.
  Note that there are 2 programming problems at the end.
  Be sure to allow enough time for these questions

Table 1.5.1: Laws of propositional logic.

Idempotent laws:	$p \vee p \equiv p$	$p \wedge p \equiv p$
Associative laws:	$(p \vee q) \vee r \equiv p \vee (q \vee r)$	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$
Commutative laws:	$p \vee q \equiv q \vee p$	$p \wedge q \equiv q \wedge p$
Distributive laws:	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$
Identity laws:	$p \vee F \equiv p$	$p \wedge T \equiv p$
Domination laws:	$p \wedge F \equiv F$	$p \vee T \equiv T$
Double negation law:	$\neg\neg p \equiv p$	
Complement laws:	$p \wedge \neg p \equiv F$ $\neg T \equiv F$	$p \vee \neg p \equiv T$ $\neg F \equiv T$
De Morgan's laws:	$\neg(p \vee q) \equiv \neg p \wedge \neg q$	$\neg(p \wedge q) \equiv \neg p \vee \neg q$
Absorption laws:	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
Conditional identities:	$p \rightarrow q \equiv \neg p \vee q$	$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$

Table 1.12.1: Rules of inference known to be valid arguments.

Rule of inference	Name
p $p \rightarrow q$ $\therefore q$	Modus ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	Modus tollens
p $\therefore p \vee q$	Addition
$p \wedge q$ $\therefore p$	Simplification

Rule of inference	Name
p q $\therefore p \wedge q$	Conjunction
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	Disjunctive syllogism
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	Resolution

Table 1.13.1: Rules of inference for quantified statemen

Rule of Inference	Name
c is an element (arbitrary or particular) $\forall x\, P(x)$ $\therefore P(c)$	Universal instantiation
c is an arbitrary element $P(c)$ $\therefore \forall x\, P(x)$	Universal generalization
$\exists x\, P(x)$ $\therefore$ (c is a particular element) $\wedge P(c)$	Existential instantiation*
c is an element (arbitrary or particular) $P(c)$ $\therefore \exists x\, P(x)$	Existential generalization

Table 3.6.1: Set identities.

Name	Identities	
Idempotent laws	$A \cup A = A$	$A \cap A = A$
Associative laws	$(A \cup B) \cup C = A \cup (B \cup C)$	$(A \cap B) \cap C = A \cap (B \cap C)$
Commutative laws	$A \cup B = B \cup A$	$A \cap B = B \cap A$
Distributive laws	$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$
Identity laws	$A \cup \varnothing = A$	$A \cap U = A$
Domination laws	$A \cap \varnothing = \varnothing$	$A \cup U = U$
Double Complement law	$\overline{\overline{A}} = A$	
Complement laws	$A \cap \overline{A} = \varnothing$ $\overline{U} = \varnothing$	$A \cup \overline{A} = U$ $\overline{\varnothing} = U$
De Morgan's laws	$\overline{A \cup B} = \overline{A} \cap \overline{B}$	$\overline{A \cap B} = \overline{A} \cup \overline{B}$
Absorption laws	$A \cup (A \cap B) = A$	$A \cap (A \cup B) = A$

Part I – Theoretical:

- ***You don't need to justify your answers to the questions in this part.***
- ***For multiple choice questions, there could be more than one answer.***
- *For all questions in this part of the exam (questions 1-12), you should submit a **single** '.cpp' file. Write your answers as one long comment (`/* … */`).*
  Name this file 'YourNetID_q1to12.cpp'.

Question 1 (8 points)

a. Convert the decimal number $(185)_{10}$ to its **base-4** representation.
b. Convert the 8-bits two's complement number $(11100101)_{\text{8-bit two's complement}}$ to its decimal representation.

Question 2 (4 points)

Suppose we have two hypotheses as follows:

h1: I left my notes in the library or I finished the rough draft of the paper.
h2: I did not leave my notes in the library or I revised the bibliography.

Therefore, I finished the rough draft of the paper or I revised the bibliography.

What is the rule of inference being used in the above statement:
a. Modus ponens
b. Resolution
c. Hypothetical syllogism
d. Modus tollens
e. None of the above

Question 3 (5 points)

The domain of the variable x consists of all the students in a university, the domain of the variable y consists of all the courses offered by that university. Define the predicates:
A(x): x is a part-time student.
F(x): x is a freshman student.
U(y): y is an upper-level course.
T(x, y): student x is taking course y.

Select the logical expression that is equivalent to: "Every part-time freshman student is taking some upper-level course."
a. $\forall x \exists y\ [(A(x) \wedge F(x)) \wedge (U(y) \wedge T(x,y))]$
b. $\forall x \exists y\ [(A(x) \wedge F(x)) \rightarrow (U(y) \wedge T(x,y))]$
c. $\forall x \forall y\ [(A(x) \wedge F(x)) \rightarrow (U(y) \wedge T(x,y))]$
d. $\forall x \exists y\ [(A(x) \wedge F(x)) \vee (U(y) \wedge T(x,y))]$
e. None of the above

Question 4 (5 points)
Select the propositions that are logically equivalent to $\neg((\neg q \to \neg p) \land (q \to \neg p))$.

a. q
b. $\neg p \lor q$
c. $\neg p$
d. p
e. None of the above

Question 5 (5 points)
Suppose you want to prove a theorem of the form "**if $\neg q$ then $\neg p$**". If you give a proof by contraposition, what do you assume and what do you prove?

a. Assume $\neg p$ is true, prove that ¬q is true.
b. Assume p is true, prove that q is true.
c. Assume $\neg q$ is true, prove that ¬p is true.
d. Assume $(\neg p \lor q)$ is true, prove that q is true.
e. None of the above

Question 6 (5 points)
Select the logical expressions that is equivalent to: $\neg\forall y \exists x \exists z\, (P(x,y,z) \land \neg Q(x,y))$

a. $\exists y \exists x\, \neg\exists z\, (P(x,y,z) \lor Q(x,y))$
b. $\exists y \forall x \forall z\, (\neg P(x,y,z) \land Q(x,y))$
c. $\exists y \forall x \forall z\, (\neg P(x,y,z) \lor Q(x,y)$
d. $\exists y \forall x \forall z\, (P(x,y,z) \land \neg Q(x,y))$
e. None of the above

Question 7 (4 points)
If A, B, and C are sets such that $(A \cap C) = (B \cap C)$ and $(A \cup C) = (B \cup C)$.
Select the statements that are **true.**

a. $A \neq B$
b. $\mathrm{B} \subseteq A$
c. $\mathrm{A} \subseteq B$
d. $A = B$
e. None of the above

Question 8 (4 points)
Let f be the function from the set of real numbers to the set of real numbers with $f(x) = 3x - 5$.
Select the statements that are **true.**

a. f(x) is not invertible.
b. $f^{-1}(x) = \frac{x+5}{3}$
c. f(x) is both one to one and onto function.
d. f(x) is one to one function but not onto function.
e. None of the above

Question 9 (10 points)
$A = \{a, b, c, d, \{a\}, \{a, b\}, \{c\}, \{a, b, c\}\}$.
For each of the following statements, state if they are true or false (no need to explain your choice).
a. $\emptyset \subseteq P\,(A)$
b. $\{a, b\} \subseteq A$
c. $\{a, b, c\} \in A$
d. $\{\{a\}\} \in A$
e. $(\{a, b\}, \{a, b, c\}) \in A \times A$
f. $\{a, b, \{a\}\} \subseteq P(A)$
g. $\{a, b, c, \{a\}\} \subseteq A$
h. $|\emptyset| = 1$
i. $\emptyset \subseteq A$
j. $|P(A)| = 8$

Question 10 (5 points)
Let A be defined to be the set $\{1, 2, 3\}$.

Let f be a function: $f: \mathrm{P}(A) \rightarrow P(A)$, defined as follows:

for X ⊆ A, f(X) = A-X.

Select the correct description of the function f.
a. One-to-one and onto
b. One-to-one but not onto
c. Not one-to-one but onto
d. Neither one-to-one nor onto
e. None of the above

Question 11 (4 points)
The domain and target set of functions f and g are **Z**. The functions are defined as:
$f(x) = x^2 + 1$ and $g(x) = 3x + 2$

An explicit formula for the function: $g \circ f(x)$ will be
a. $3x^2 + 6x + 7$
b. $3x^2 + 5$
c. $3x^2 + 6x + 5$
d. None of the above

Question 12 (4 points)
Select the set that is equivalent to $\overline{A} \cap (A \cup B)$.
a. $\emptyset$
b. A
c. $\overline{A} \cup B$
d. $\overline{A} \cap B$
e. None of the above

Part II – Coding:

- *For **each** question in this part (questions 13-14), you should submit a '.cpp' file, containing your code.*
- *Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose most suitable control statements, etc.*
- *In all questions, you may assume that the user enters inputs as they are asked.*
  For example, if the program expects a positive integer, you may assume that user will enter positive integers.
- *No need to document your code. However, you may add comments if you think they are needed for clarity.*

Question 13 (17 points)

Write a C++ program that reads a positive odd integer, n, and prints a shape of (n+1) lines consisting of asterisks (*) and spaces as follows:

1st line: print (n-1)/2 spaces and then print 1 asterisk
2nd line: print 0 space and n asterisks
3rd line: print (n-3)/2 spaces and then print 3 asterisks
4th line: print 0 space and n asterisks
5th line: print (n-5)/2 spaces and then print 5 asterisks
6th line: print 0 space and n asterisks
…
…
…
n-th line: print 0 space and n asterisks
(n+1)th line: print 0 space and n asterisks

Your program should interact with the user **exactly** as demonstrated in the following three executions (color is used just for the illustration purpose only):

Execution example 1:

```
Please enter a positive integer:
3
 *
***
***
***
```

<u>Execution example 2</u>:

```
Please enter a positive integer:
5
  *
*****
 ***
*****
*****
*****
```

<u>Execution example 3</u>:

```
Please enter a positive integer:
7
   *
*******
  ***
*******
 *****
*******
*******
*******
```

<u>Execution example 4</u>:

```
Please enter a positive integer:
1
*
*
```

<u>Execution example 5</u>:

```
Please enter a positive integer:
11
     *
***********
    ***
***********
   *****
***********
  *******
***********
 *********
***********
***********
***********
```

Question 14 (20 points)

A sequence of positive numbers has been given. Each of these positive numbers represents a student's score in an Exam and will be in the range [1, 100]. Suppose we define grading Categories as follows:

Excellent: Students who got scores greater than or equal to 90.
Very Good: Students who got scores greater than or equal to 80 but less than 90.
Average: Students who got scores greater than or equal to 70 but less than 80.
Below Average: Students who got scores less than 70.

Write a C++ program that reads from the user a sequence of scores (positive numbers in between 1 and 100) and prints **the following statistics**.

Number of students in the **Excellent** Category:
Number of students in the **Very Good** Category:
Number of students in the **Average** Category:
Number of students in the **Below Average** Category:
Total number of students in the given sequence of scores:
Maximum score among the given sequence of scores:
Minimum score among the given sequence of scores:
Overall Average score of the given sequence of scores:

Implementation requirement:

a. **The user should enter their numbers, each one in a separate line, and type -1 to indicate the end of the input.**
b. You are not allowed to use C++ syntactic features that were not covered in the Bridge program so far.
c. You are not allowed to use any **cmath** or **math.h** library function for this program. You have to calculate maximum, minimum, and average scores without using any library function.
d. While printing overall average score, you can just print the raw value of average score. That is, you don't need to format it (like showing two digits after decimal point).

Your program should interact with the user **exactly** the same way, as demonstrated in the following two executions:

<u>Execution example 1</u>:

```
Please enter a sequence of numbers in between 1 and 100, each one in a
separate line. End your sequence by typing -1:
94
60
96
88
34
86
67
61
79
81
96
89
77
99
77
86
-1
Number of students in the Excellent Category: 4
Number of students in the Very Good Category: 5
Number of students in the Average Category: 3
Number of students in the Below Average Category: 4
Total number of people in the given sequence of scores: 16
Maximum score among the given sequence of scores: 99
Minimum score among the given sequence of scores: 34
Overall Average score of the given sequence of scores: 79.375
```

Execution example 2:

```
Please enter a sequence of numbers in between 1 and 100, each one in a
separate line. End your sequence by typing -1:
78
59
73
97
71
70
90
85
59
86
90
91
66
77
73
54
45
86
86
77
45
89
-1
Number of students in the Excellent Category: 4
Number of students in the Very Good Category: 5
Number of students in the Average Category: 7
Number of students in the Below Average Category: 6
Total number of people in the given sequence of scores: 22
Maximum score among the given sequence of scores: 97
Minimum score among the given sequence of scores: 45
Overall Average score of the given sequence of scores: 74.8636
```