NYU, Tandon School of Engineering
CS Bridge — Summer 2023

Exam 1

Thursday, August 10, 2023

- 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:	$p \lor p \equiv p$	$p \land p \equiv p$
Associative laws:	$(p \lor q) \lor r \equiv p \lor (q \lor r)$	$(p \land q) \land r \equiv p \land (q \land r)$
Commutative laws:	$p \lor q \equiv q \lor p$	$p \land q \equiv q \land p$
Distributive laws:	$p \lor (q \land r) \equiv (p \lor q) \land (p \lor r)$	$p \land (q \lor r) \equiv (p \land q) \lor (p \land r)$
Identity laws:	$p \lor F \equiv p$	$p \land T \equiv p$
Domination laws:	$p \land F \equiv F$	$p \lor T \equiv T$
Double negation law:	$\neg\neg p \equiv p$	
Complement laws:	$p \land \neg p \equiv F$ $\neg T \equiv F$	$p \lor \neg p \equiv T$ $\neg F \equiv T$
De Morgan's laws:	$\neg(p \lor q) \equiv \neg p \land \neg q$	$\neg(p \land q) \equiv \neg p \lor \neg q$
Absorption laws:	$p \lor (p \land q) \equiv p$	$p \land (p \lor q) \equiv p$
Conditional identities:	$p \to q \equiv \neg p \lor q$	$p \leftrightarrow q \equiv (p \to q) \land (q \to p)$

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

Rule of inference	Name
p $p \to q$ $\therefore q$	Modus ponens
$\neg q$ $p \to q$ $\therefore \neg p$	Modus tollens
p $\therefore p \lor q$	Addition
$p \land q$ $\therefore p$	Simplification

Rule of inference	Name
p q $\therefore p \land q$	Conjunction
$p \to q$ $q \to r$ $\therefore p \to r$	Hypothetical syllogism
$p \lor q$ $\neg p$ $\therefore q$	Disjunctive syllogism
$p \lor q$ $\neg p \lor r$ $\therefore q \lor r$	Resolution

Table 1.13.1: Rules of inference for quantified stateme

Rule of Inference	Name
c is an element (arbitrary or particular) ∀x P(x) ∴ P(c)	Universal instantiation
c is an arbitrary element P(c) ∴ ∀x P(x)	Universal generalization
∃x P(x) ∴ (c is a particular element) ∧ P(c)	Existential instantiation*
c is an element (arbitrary or particular) P(c) ∴ ∃x P(x)	Existential generalization

Table 3.6.1: Set identities.

Name	Identities	
Idempotent laws	A ∪ A = A	A ∩ A = A
Associative laws	(A ∪ B) ∪ C = A ∪ (B ∪ C)	(A ∩ B) ∩ C = A ∩ (B ∩ C)
Commutative laws	A ∪ B = B ∪ A	A ∩ B = B ∩ A
Distributive laws	A ∪ (B ∩ C) = (A ∪ B) ∩ (A ∪ C)	A ∩ (B ∪ C) = (A ∩ B) ∪ (A ∩ C)
Identity laws	A ∪ ∅ = A	A ∩ *U* = A
Domination laws	A ∩ ∅ = ∅	A ∪ *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 ∪ (A ∩ B) = A	A ∩ (A ∪ 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 $(4859)_{10}$ to its **base-5** representation.
b. Convert the 8-bits two's complement number $(11000111)_{\text{8-bit two's complement}}$ to its decimal representation.

Question 2 (4 points)

Select the propositions that are logically equivalent to $\neg((\neg q \rightarrow p) \wedge (q \rightarrow p))$.

a. q
b. $\neg p \vee q$
c. $\neg p$
d. p
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.
M(y): y is a math course.
T(x, y): student x is taking course y.

Select the logical expression that is equivalent to: "There is a part-time student who is not taking any math course."

a. $\forall x \exists y\, [A(x) \wedge (M(y) \rightarrow \neg\, T(x,y))]$
b. $\exists x \forall y\, [A(x) \wedge (M(y) \rightarrow T(x,y))]$
c. $\exists x \forall y\, [A(x) \wedge (M(y) \rightarrow \neg\, T(x,y))]$
d. $\exists x \forall y\, [A(x) \rightarrow (M(y) \rightarrow \neg\, T(x,y))]$
e. None of the above

Question 4 (5 points)
Show that at least four of any 37 days chosen must fall on the same month of the year.

A proof by contradiction of the above statement starts by assuming which fact?
a. Suppose that at least three of the 37 days fall on the same month of the year.
b. Suppose that at most four of the 37 days fall on the same month of the year.
c. Suppose that exactly four of the 37 days fall on the same month of the year.
d. Suppose that at most three of the 37 days fall on the same month of the year.
e. None of the above

Question 5 (5 points)
Select the logical expressions that is equivalent to: $\forall y \exists x \exists z\,(\neg P(x,y,z) \rightarrow \neg Q(x,y))$
a. $\forall y \exists x\, \neg \forall z\,(P(x,y,z) \vee Q(x,y))$
b. $\exists y \forall x \forall z\,(\neg P(x,y,z) \wedge Q(x,y))$
c. $\forall y \neg \forall x \forall z\,(\neg P(x,y,z) \wedge Q(x,y)$
d. $\exists y \forall x \forall z\,(P(x,y,z) \wedge \neg Q(x,y))$
e. None of the above

Question 6 (5 points)
Determine whether the following set is the power set of some set. If the following set is a power set, give the set of which it is a power set.

$$\{\emptyset, \{\emptyset\}, \{\mathbf{1}\}, \{\boldsymbol{a}\}\ \{\emptyset, \mathbf{1}\}, \{\mathbf{a}, \mathbf{1}\}, \{\emptyset, \boldsymbol{a}\}, \{\emptyset, \boldsymbol{a}, \mathbf{1}\}\}$$

a. No, this set is not a power set of any set.
b. Yes, and the set is $\{\{\emptyset\}, \mathbf{1}, \boldsymbol{a}\}$
c. Yes, and the set is $\{\emptyset, \mathbf{1}, \boldsymbol{a}\}$
d. Yes, and the set is $\{\mathbf{1}, \boldsymbol{a}\}$
e. Can't be determined

Question 7 (10 points)
$A = \{a, b, 3, 4, \{a\}, \{3\}, \{a, b, 3\}\}$.
For each of the following statements, state if they are true or false (no need to explain your choice).
a. $\emptyset \subseteq A$
b. $\{4\} \subseteq A$
c. $\{a, b, 3\} \in A$
d. $\{a, b, 3\} \subseteq A$
e. $\{\{3\}\} \in A$
f. $(b, \{a, b, 3\}) \in A \times A$
g. $\{a, b, 3,\ \{a\}\} \subseteq P(A)$
h. $\{a, b, 3,\ \{a\}\} \subseteq A$
i. $|\emptyset| = 0$
j. $\{\emptyset\} \subseteq P(A)$

Question 8 (4 points)

Select the set that is equivalent to $\overline{A} \cap (\overline{B} \cup A)$.

a. $\emptyset$
b. U
c. $\overline{A} \cup \overline{B}$
d. $\overline{A} \cap \overline{B}$
e. None of the above

Question 9 (5 points)

Let M be defined to be the set $\{1, 2, 3\}$. Let f be a function: $f: P(M) \to P(M)$, defined as follows:

for X ∈ $P(M)$, f(X) = M ∩ 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 10 (4 points)

The domain and target set of functions f and g are **Z**. The functions are defined as:

$f(x) = x + 1$ and $g(x) = 3x^2 + 2$

An explicit formula for the function: f o g(x) will be

a. $3x^2 + 6x + 7$
b. $3x^2 + 3$
c. $3x^2 + 6x + 5$
d. None of the above

Question 11 (4 points)

Let f be the function from the set of integers to the set of integers with f(x) = 2x+1. Select the statements that are **true.**

a. $f^{-1}(x) = (x-1)/2$.
b. f(x) is not invertible.
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 12 (4 points)

If it snows today, the university will be closed. The university will not be closed today. Therefore, it did not snow today.

What is the rule of inference being used in the above statement:

a. Modus ponens
b. Disjunctive syllogism
c. Hypothetical syllogism
d. Modus tollens

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 integer, n, and prints a shape of (2*n+1) lines consisting of asterisks (*) and spaces as follows:

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

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:
6
      *
     ***
    *****
   *******
  *********
 ***********
*************
 ***********
  *********
   *******
    *****
     ***
      *
```

Question 14 (20 points)

A sequence of positive integers has been given. Each of these positive integers will have at least 1 digit and at most 9 digits. The first digit of these integers will not be 0 (Zero) and each of the integers will be greater than 1. Suppose we define **Perfect Square** and **Perfect Number** as follows:

Perfect Square: A **perfect square** is a positive integer that is obtained by multiplying an integer by itself. For example, 16 is a perfect square because it is the product of integer 4 by itself, 4 × 4 = 16. However, 20 is not a perfect square number because it cannot be expressed as the product of two same integers.

Perfect Number: A **perfect number** is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3 (excluding itself), and 1 + 2 + 3 = 6, so 6 is a perfect number. Similarly, 28 has divisors 1, 2, 4, 7 and 14 (excluding itself), and 1 + 2 + 4 + 7+ 14 = 28, so 28 is a perfect number.

Write a C++ program that reads from the user a sequence of integers (positive integers with at least 1-digit and at most 9 digits and each integer will be greater than 1) and prints the following statistics.

Total count of Perfect Squares in the given sequence:
Total count of Perfect Numbers in the given sequence:

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. If you need, you can use **sqrt() function as defined in the cmath header file.** You are not allowed to use any other **cmath** or **math.h** library function for this program.
d. The first digit of these integers will not be 0 (Zero). Each of the positive integers will have at least 1-digit and at most 9 digits and each of the positive integers will be greater than 1.

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

Execution example 1:

```
Please enter a sequence of integers (with at least 1-digit and at most 9-
digits), each one in a separate line. End your sequence by typing -1:
6
9
4
2
324
28
87
81
64
36
6
12345
1234
144
400
8128
90
496
28
49
525
10000
678
29
-1
Total count of Perfect Squares in the given sequence: 10
Total count of Perfect Numbers in the given sequence: 6
```

<u>Execution example 2</u>:

```
Please enter a sequence of integers (with at least 1-digit and at most 9-
digits), each one in a separate line. End your sequence by typing -1:
2
33
98
6
6
6
9
8
16
25
49
81
121
456
83726261
30
81
28
221
496
76
8128
28
496
873
891
112
25
55
1293
-1
Total count of Perfect Squares in the given sequence: 8
Total count of Perfect Numbers in the given sequence: 8
```