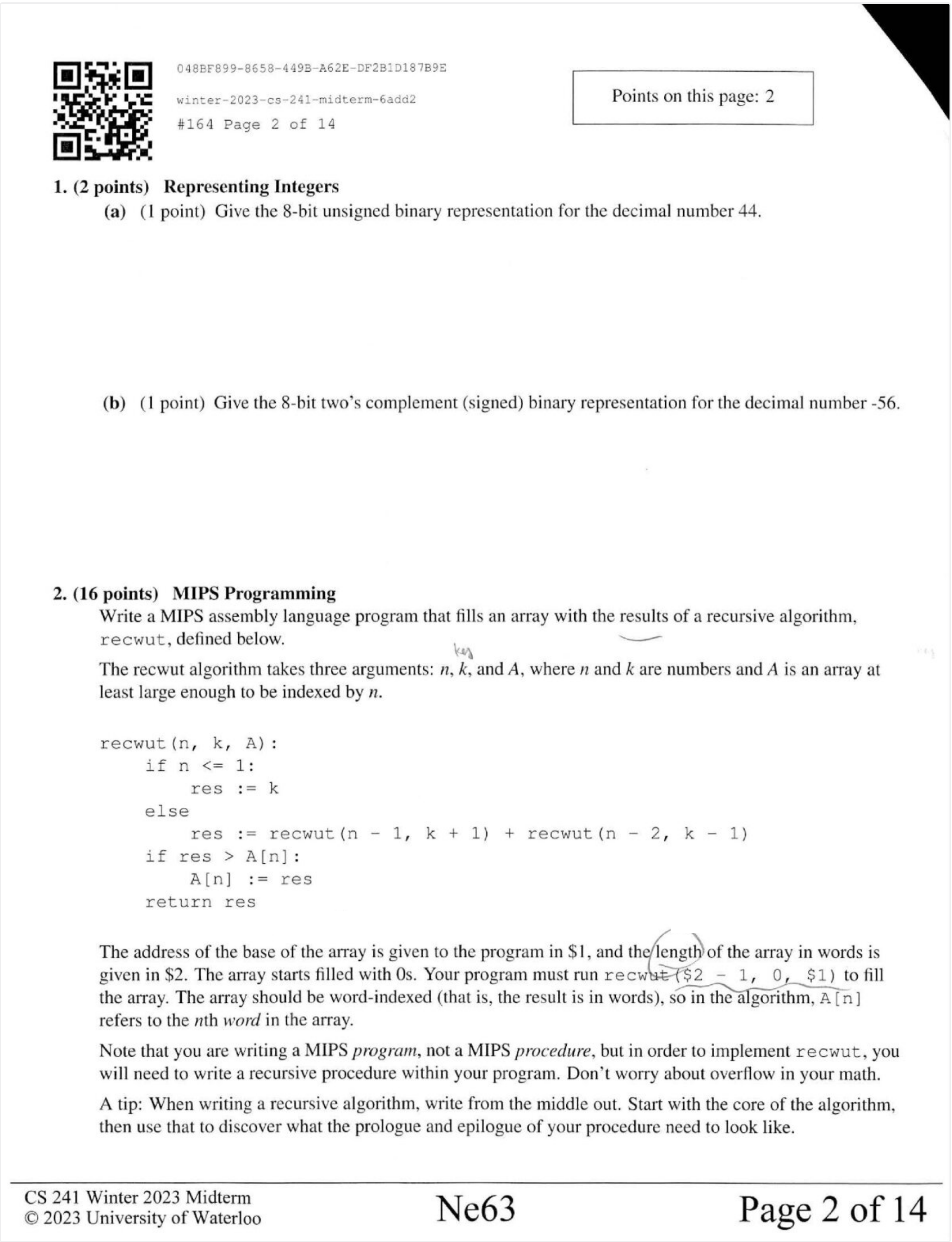
My grades for **Winter 2023 CS 241 Midterm**

Q1 2



048BF899-8658-449B-A62E-DF2B1D187B9E

winter-2023-cs-241-midterm-6add2

#164 Page 2 of 14

Points on this page: 2

1. (2 points) Representing Integers

(a) (1 point) Give the 8-bit unsigned binary representation for the decimal number 44.

(b) (1 point) Give the 8-bit two's complement (signed) binary representation for the decimal number -56.

2. (16 points) MIPS Programming

Write a MIPS assembly language program that fills an array with the results of a recursive algorithm, `recwut`, defined below.

key

The recwut algorithm takes three arguments: *n*, *k*, and *A*, where *n* and *k* are numbers and *A* is an array at least large enough to be indexed by *n*.

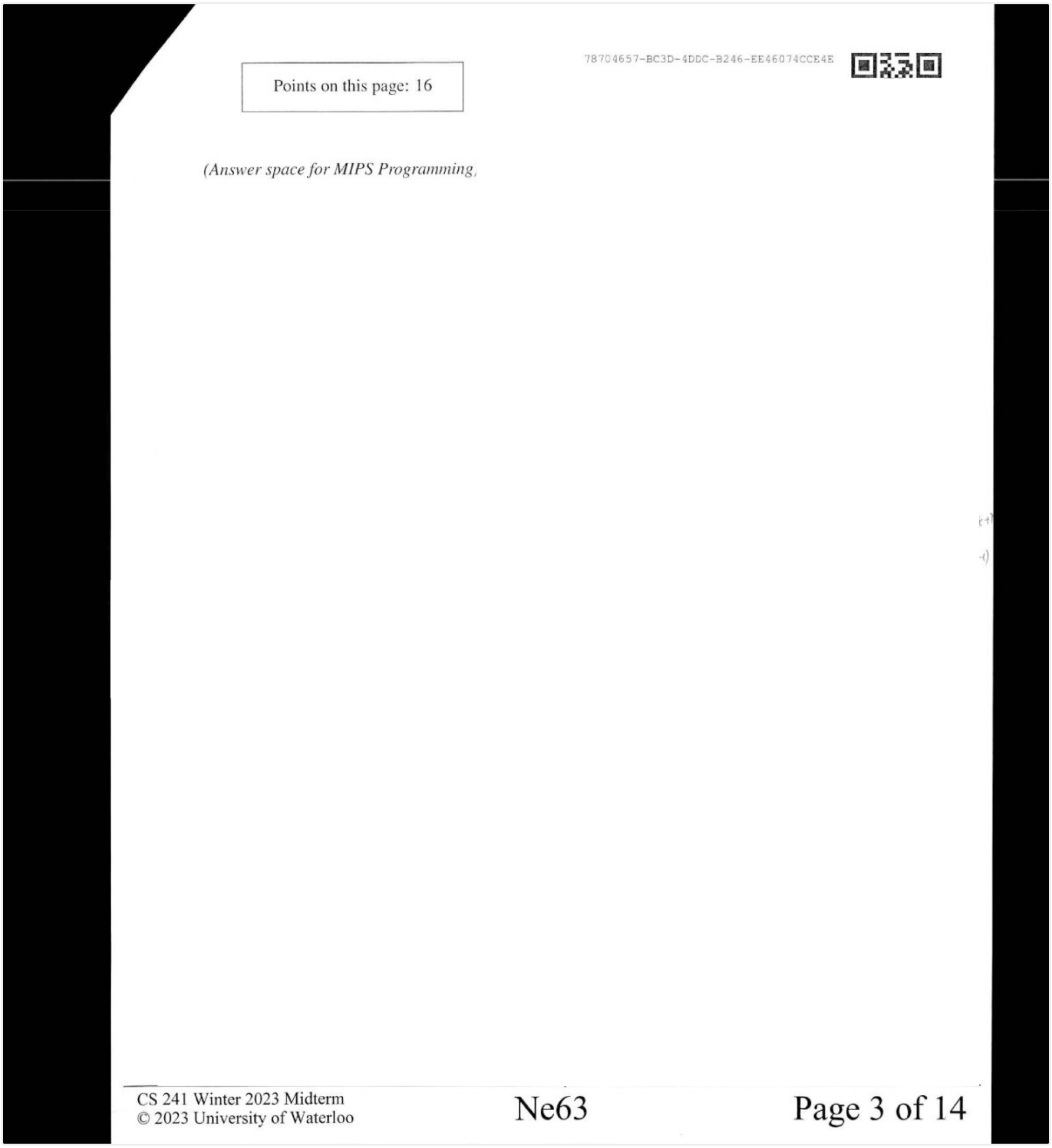
```
recwut(n, k, A):
    if n <= 1:
        res := k
    else
        res := recwut(n - 1, k + 1) + recwut(n - 2, k - 1)
    if res > A[n]:
        A[n] := res
    return res
```

The address of the base of the array is given to the program in $1, and the length of the array in words is given in $2. The array starts filled with 0s. Your program must run `recwut($2 - 1, 0, $1)` to fill the array. The array should be word-indexed (that is, the result is in words), so in the algorithm, `A[n]` refers to the *n*th *word* in the array.

Note that you are writing a MIPS *program*, not a MIPS *procedure*, but in order to implement `recwut`, you will need to write a recursive procedure within your program. Don't worry about overflow in your math.

A tip: When writing a recursive algorithm, write from the middle out. Start with the core of the algorithm, then use that to discover what the prologue and epilogue of your procedure need to look like.

Q2 15



Points on this page: 16

78704657-BC3D-4DDC-B246-EE46074CCE4E

(Answer space for MIPS Programming,

Points on this page: 3

Q3abc 3

3. (9 points) Assembly Language and Machine Code

Consider the following MIPS assembly language program:

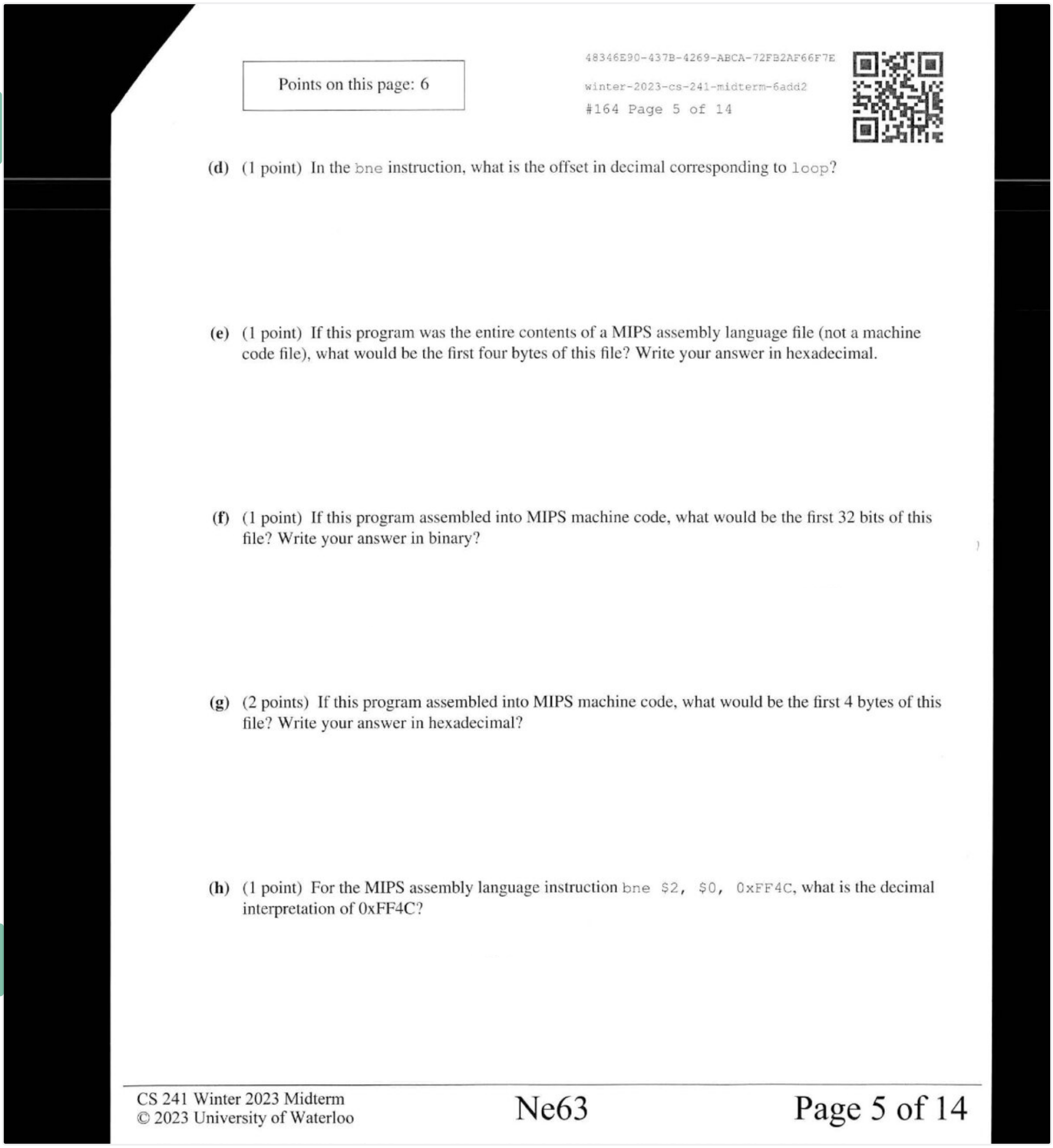
```
AbsSum:    ; $1 start addr of array
           ; $2 size of array
           ; $3  answer

loop:      lw $6, 0($1)
           slt $7, $6, $0          ;if A[i] < 0
           beq $7, $0, endif
           sub $6, $0, $6          ; abs value
endif:
           add $3, $3, $6
           add $5,$5,$4            ; $5 = $5 + 4
           sub $2, $2, $8          ; n -=1;
           bne $2, $0, loop        ; loop until $2==0

end:    jr $31
```

16.

(a) (1 point) In the symbol table, what is the address in hexadecimal corresponding to `endif`?

(b) (1 point) In the symbol table, what is the address in hexadecimal corresponding to `end`?

(c) (1 point) In the `beq` instruction, what is the offset in decimal corresponding to `endif`?



Q3de 2

(d) (1 point) In the `bne` instruction, what is the offset in decimal corresponding to `loop`?

(e) (1 point) If this program was the entire contents of a MIPS assembly language file (not a machine code file), what would be the first four bytes of this file? Write your answer in hexadecimal.

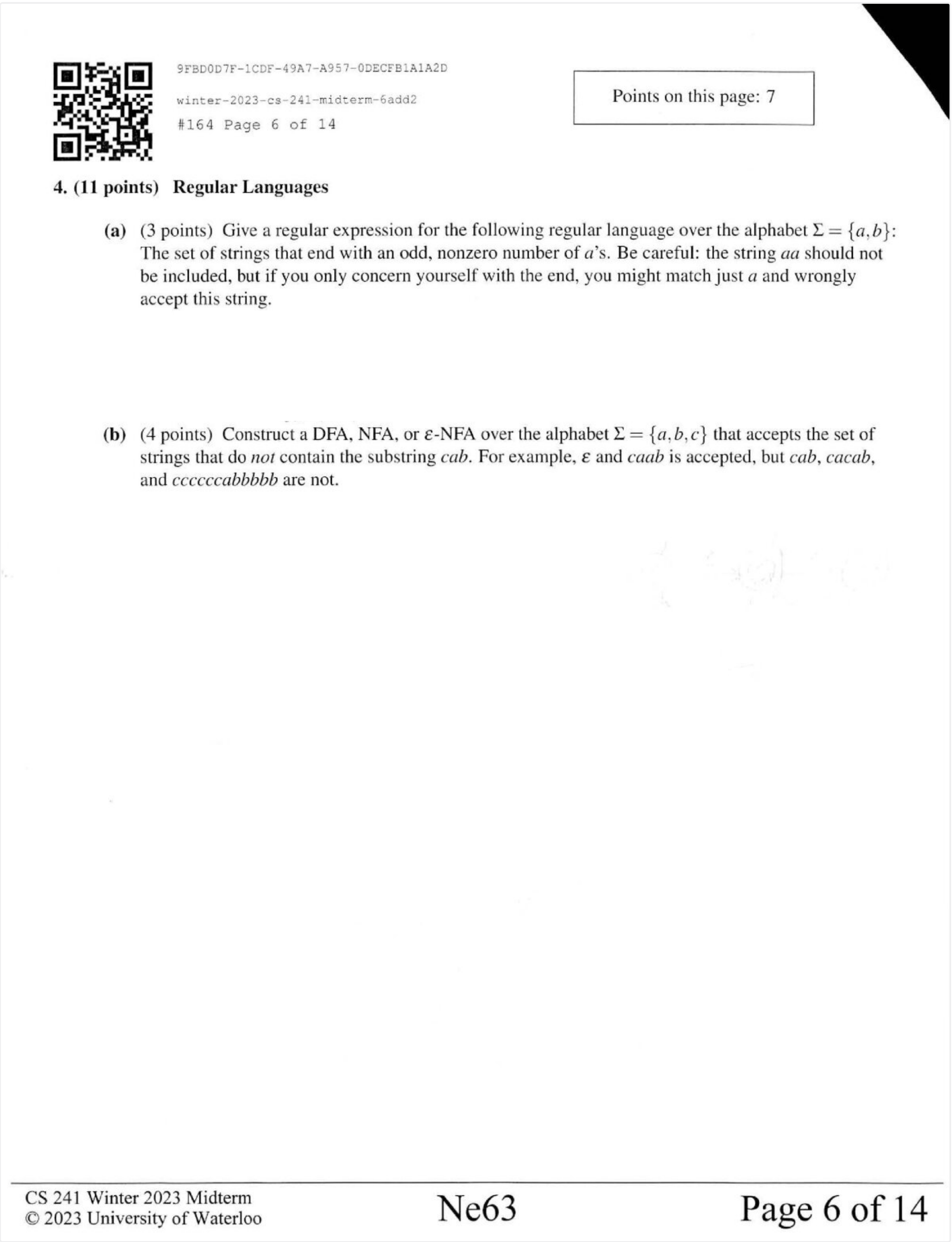
Q3fg 3

(f) (1 point) If this program assembled into MIPS machine code, what would be the first 32 bits of this file? Write your answer in binary?

(g) (2 points) If this program assembled into MIPS machine code, what would be the first 4 bytes of this file? Write your answer in hexadecimal?

Q3h 1

(h) (1 point) For the MIPS assembly language instruction `bne $2, $0, 0xFF4C`, what is the decimal interpretation of 0xFF4C?



Points on this page: 7

Q4a 3

4. (11 points) Regular Languages

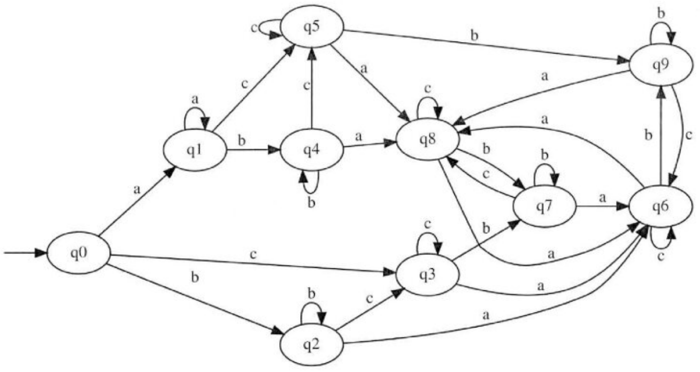
(a) (3 points) Give a regular expression for the following regular language over the alphabet $\Sigma = \{a, b\}$: The set of strings that end with an odd, nonzero number of a's. Be careful: the string *aa* should not be included, but if you only concern yourself with the end, you might match just a and wrongly accept this string.

Q4b 4

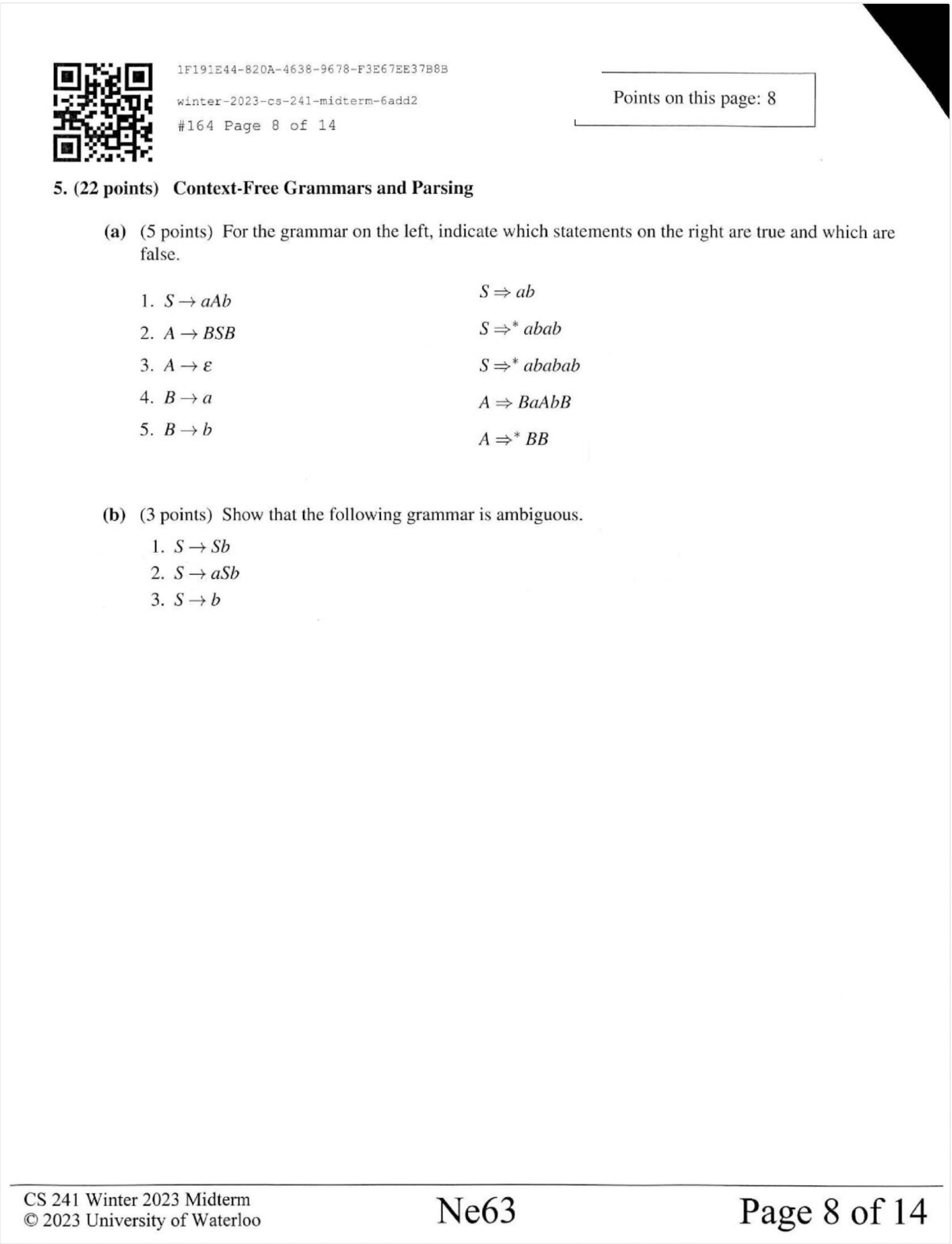
(b) (4 points) Construct a DFA, NFA, or ε-NFA over the alphabet $\Sigma = \{a, b, c\}$ that accepts the set of strings that do *not* contain the substring *cab*. For example, ε and *caab* is accepted, but *cab*, *cacab*, and *cccccabbbbb* are not.

Q4c 4

(c) (4 points) The following DFA over the alphabet $\{a, b, c\}$ has no accepting states. For each of the following descriptions of languages, give the states in the DFA that would need to be accepting for it to accept the described language.



Language	Accepting states
Even number of *a*s (any number of *b*s and *c*s)	
Odd number of *a*s (any number of *b*s and *c*s)	
Ends with *b*	
Letters in alphabetical order	
$\{\varepsilon\}$	
{} (empty language)	



Points on this page: 8

Q5a 4

5. (22 points) **Context-Free Grammars and Parsing**

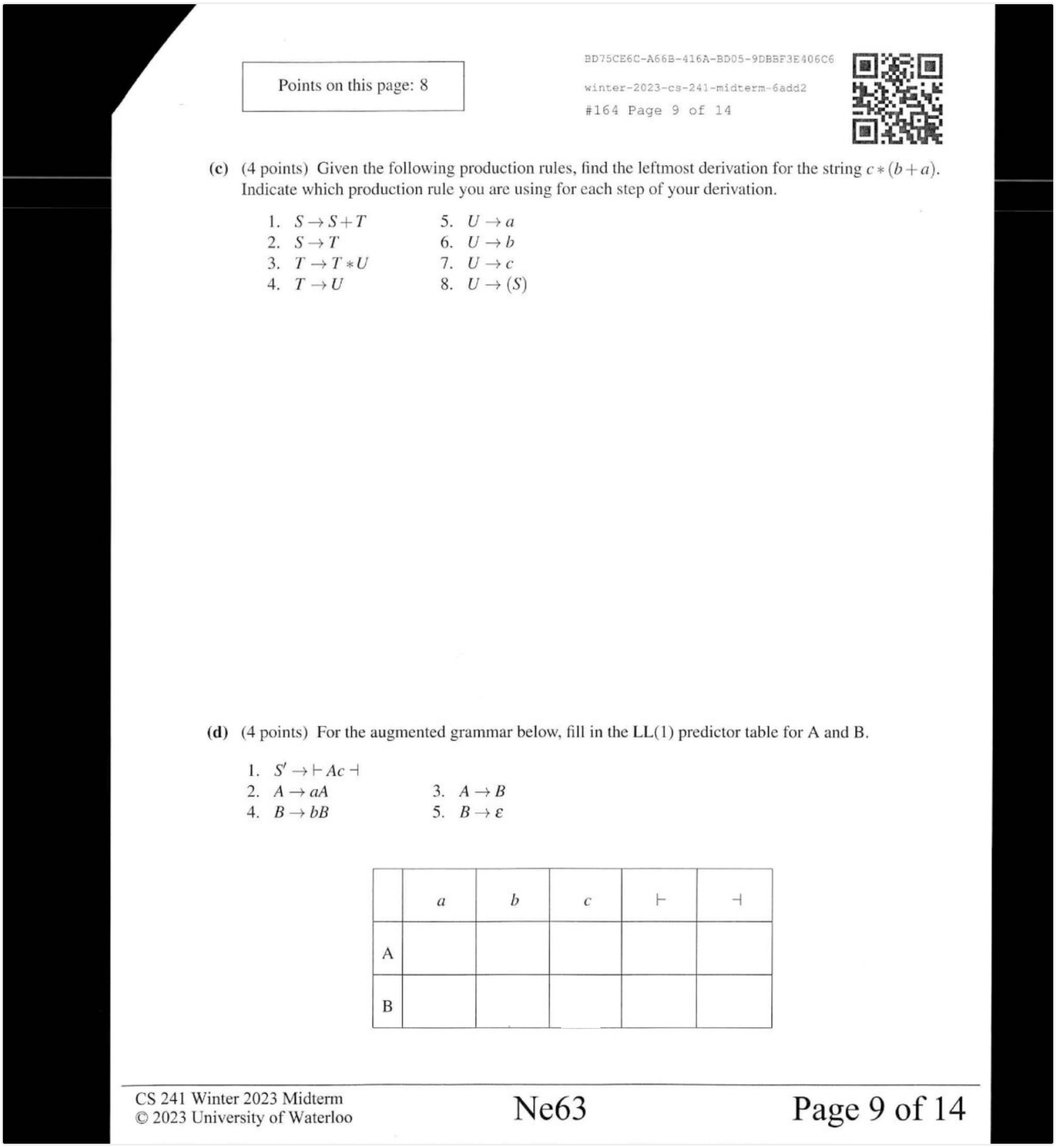
(a) (5 points) For the grammar on the left, indicate which statements on the right are true and which are false.

1. $S \to aAb$	$S \Rightarrow ab$
2. $A \to BSB$	$S \Rightarrow^* abab$
3. $A \to \varepsilon$	$S \Rightarrow^* ababab$
4. $B \to a$	$A \Rightarrow BaAbB$
5. $B \to b$	$A \Rightarrow^* BB$

Q5b 3

(b) (3 points) Show that the following grammar is ambiguous.

1. $S \to Sb$
2. $S \to aSb$
3. $S \to b$



Points on this page: 8

Q5c 4

(c) (4 points) Given the following production rules, find the leftmost derivation for the string $c*(b+a)$. Indicate which production rule you are using for each step of your derivation.

1. $S \rightarrow S+T$
2. $S \rightarrow T$
3. $T \rightarrow T*U$
4. $T \rightarrow U$
5. $U \rightarrow a$
6. $U \rightarrow b$
7. $U \rightarrow c$
8. $U \rightarrow (S)$

Q5d 4

(d) (4 points) For the augmented grammar below, fill in the LL(1) predictor table for A and B.

1. $S' \rightarrow \vdash Ac \dashv$
2. $A \rightarrow aA$
3. $A \rightarrow B$
4. $B \rightarrow bB$
5. $B \rightarrow \varepsilon$

	a	b	c	$\vdash$	$\dashv$
A					
B					

Points on this page: 6

Q5e 6

(e) (6 points) Given the following grammar and predict table, fill in the next six lines of the LL(1) parse table (i.e. just the beginning of the derivation) for the input $\vdash abc \dashv$. For each row, indicate the read input, the unread input (a.k.a. what you are processing), the contents of the stack, and the action you would take next.

1. $S' \to \vdash S \dashv$ 2. $S \to AC$
3. $A \to aA$ 4. $A \to b$
5. $C \to aC$ 6. $C \to c$

	a	b	c	⊢	⊣
S'				1	
S	2	2			
A	3	4			
C	5		6		

Read Input	Unread Input	Stack	Action
	$\vdash abc \dashv$	S'	

✓

Q6abc 3

adjust1 3

Points on this page: 6

6. (14 points) MIPS Logic

Each of the following MIPS programs is placed in the file `program.asm`, assembled and executed as follows:

```
cs241.binasm < program.asm > program.mips
mips.twoints program.mips
```

The input will always be two signed integers placed in $1 and $2 and the result will always be placed in $3. Circle the option that describes the results when the program is executed. If the program has errors detected in assembly, does not terminate, crashes, or does not fit in one of the first four options then pick the option *None of these*.

```
    add $3, $1, $0
    slt $4, $1, $2
    beq $4, $0, a
    add $3, $2, $0
a:  jr $31
```

Options: $3=max($1,$2) $3=min($1,$2) $3=$1 $3=$2 *None of these.*

```
    add $3, $1, $0
    slt $4, $2, $1
    bne $4, $0, a
    add $3, $2, $0
a:  jr $31
```

Options: $3=max($1,$2) $3=min($1,$2) $3=$1 $3=$2 *None of these.*

```
    slt $4, $1, $2
    beq $4, $0, a
    add $3, $1, $0
a:  slt $4, $2, $1
    beq $4, $0, b
    add $3, $2, $0
b:  jr $31
```

Options: $3=max($1,$2) $3=min($1,$2) $3=$1 $3=$2 *None of these*

Q6defg 2

adjust2 2

Points on this page: 8

```
     slt $4, $1, $2
     bne $4, $0, a
     beq $4, $0, b
a:   add $3, $1, $0
     lis $5
     jr $31
b:   add $3, $2, $0
     jr $31
```

Options: $3=max($1,$2) $3=min($1,$2) $3=$1 $3=$2 *None of these.*

```
     sub $4, $2, $1
     slt $5, $4, $0
     beq $5, $0, a
     add $2, $1, $0
a:   add $3, $2, $0
     jr $31
```

Options: $3=max($1,$2) [✓ $3=min($1,$2)] $3=$1 $3=$2 *None of these.*

```
     add $3, $1, $0
     slt $4, $1, $1
     slt $5, $1, $2
     beq $4, $5, jr
     add $3, $2, $5
     add $30, $31, $4
     jr $30
jr:  jr $31
```

Options: $3=max($1,$2) $3=min($1,$2) $3=$1 $3=$2 *None of these.*

```
         slt $4, $2, $1
         bne $4, $0, a
         add $3, $1, $0
         lis $5
         .word b
         jalr $5
a:       add $3, $2, $0
b:       jr $31
```

Options: $3=max($1,$2) $3=min($1,$2) $3=$1 $3=$2 [✓ *None of these.*]