CS 111 Midterm 2

Junhui Cho

TOTAL POINTS

32.5 / 40

QUESTION 1

1 Tracing loops and method calls 4 / 6

✓ **- 2 pts part b: did not iterate over characters in each string**

QUESTION 2

2 Tracing code with references 2.5 / 4

✓ **- 1.5 pts Values of y and z should be equal after the function returns / assigning new lists to local y and z does not affect global y and z**

QUESTION 3

3 Designing a circuit 4 / 5

✓ **- 1 pts one issue in circuit**

incorrect shapes for AND and OR gates

QUESTION 4

4 Writing a function I 5 / 5

✓ **- 0 pts Correct**

QUESTION 5

5 Writing an assembly-language program 5 / 5

✓ **- 0 pts Correct**

QUESTION 6

6 Writing a function II 3 / 5

✓ **- 2 pts returns incorrect value in some/all cases**

you did not update the value of smallest

QUESTION 7

7 Tracing loops II 4 / 5

✓ **- 1 pts part b: one issue**

QUESTION 8

8 Writing a function III 5 / 5

✓ **- 0 pts Correct**

CS 111, Fall 2018 name: Junhui Cho email: jhoo@bu.edu

Midterm 2 lecture: ___ A1 (MWF 10:10) ___ B1 (MWF 12:20) ✓ C1 (TuTh)

1. What is the output of each of the following programs? (6 points)

a)

```
def mystery1(x, y):
    for a in [x, y]:
        for b in range(x, a+1):
            print(a * b)
    x = a * 4
    return x

x = 3
y = 4
y = mystery1(x, y)
print(x, y)
```

put the output below:

9
12
16
3 16

b)

```
vals = ['oh', 'boy']
x = ''      # empty string
for i in range(2):
    for j in vals[i]:
        x = j + x
        print(x)
```

put the output below:

oh
boyoh

2. What is printed by the following program? (4 points)

```
def mystery2(x, y, z):
    x += 2
    for i in range(len(y)):
        y[i] *= 2
    print(x, y, z)
    z = [3, 6]
    print(x, y, z)

x = 3
y = [2, 4]
z = y
mystery2(x, y, z)
print(x, y, z)
```

put the output below:

5 [4, 8] [4, 8]
5 [4, 8] [3, 6]
3 [4, 8] [3, 6]

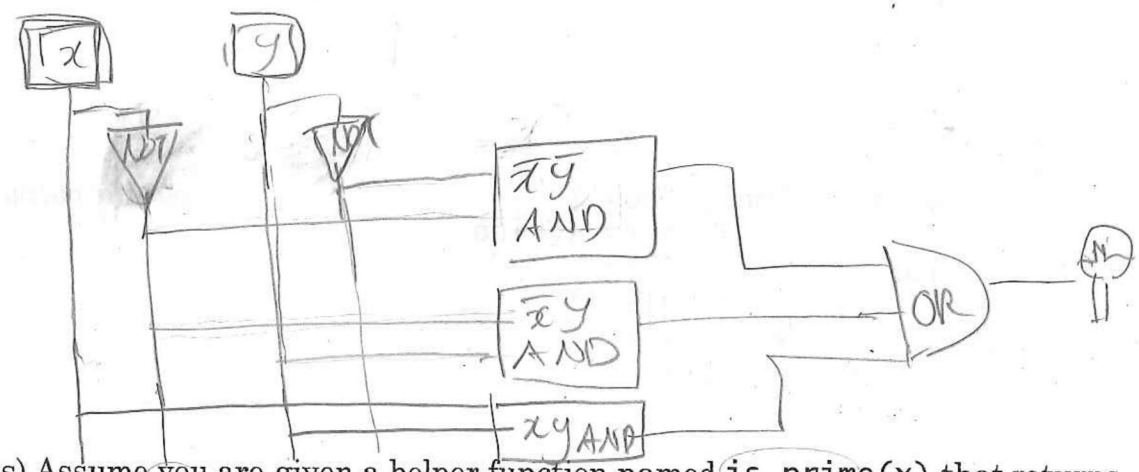
3. (5 points) a) Design a truth table for a function that takes 2 bits of input, x and y, and that produces a true output if and only if x is **less than or equal to** y.

x	y	f(x,y)
0	0	1
0	1	1
1	0	0
1	1	1

b) Determine the minterm expansion formula for your truth table from part a.

$\bar{x}\bar{y} + \bar{x}y + xy$

c) Draw the corresponding circuit. **Label** each AND gate with the minterm it represents.



4. (5 points) Assume you are given a helper function named `is_prime(x)` that returns `True` if its input `x` is prime and False otherwise. Write a function `no_primes(vals)` that returns `True` if the list of integers `vals` contains no prime numbers, and `False` otherwise. You may assume the list has at least one element. For example:

- `no_primes([6, 4, 9])` should return `True` because *none* of the values are prime.
- `no_primes([4, 5, 8])` should return `False` because the 5 is prime.

You must use a loop and the function `is_prime()`, which you do **not** need to write.

```
def no_primes(vals):
    count = 0
    for x in vals:
        if is_prime(x) == True:
            count += 1
    if count == 0:
        return True

    else:
        return False
```

email
jh00@bu.edu

5. (5 points) Write an Hmmm assembly-language program that gets **two** numbers from the user and calls a **function** that finds and returns the **larger** of the two numbers. **After** returning from the function, the program should print the larger number. You must use `calln` and `jumpr` instructions to implement your function.

```
00  read r1        # takes first input
01  read r2        # takes second input
02  calln r14 5    # call a function
03  write r13      # print r13
04  halt
05  sub r13 r1 r2  # r13 = r1-r2
06  jltz r13 9     # jump to 9 if r2>r1
07  jgtz r13 11    # jump to 11 if r1>r2
08  jeqz r13 11    # jump to 11 if r1=r2
09  copy r13 r2    # r13 = r2
10  jumpr r14      # ends the function
11  copy r13 r1 # r13 = r1
12  jumpr r14  # ends function
```

6. (5 points) Given two lists of integers that have the same length, you can calculate the products of the values in corresponding positions in the two lists. Write a function `index_min_product(list1, list2)` that returns the *index* of the *smallest* product of a value in `list1` multiplied by the corresponding value in `list2`. For example, if you are given the following lists:

```
list1 = [8, 6, 3, 2]
list2 = [7, 5, 1, 4]
```

`3*1` is the smallest product, so your function would return 2 (the index of 3 and 1). **You must use a loop!** You may assume the lists have the same non-zero length.

```
def index_min_product(list1, list2):
    min_product = list1[0] * list2*[0]
    smallest_index = 0
    for x in range(1, len(list1)):
        if list1[x]*list2[x] < min_product:
            smallest_index = x

    return smallest_index
```

7. (5 points) What is the output of the following programs?

a)

```
x = 1
while x < 8:
    x += 3
    print(x)
```

put the output below:

4
7
10

b)

```
a = 1
b = 2
while b < 10:
    if a < b:
        print('b =', b)
        a = a + b
    else:
        print('a =', a)
        a = a - 1
    b += 2
```

put the output below:

b = 2
b = 4
a = 7
b = 8

a	b
1	2
3	4
7	6
6	8
14	10

8. (5 points) Write a function `index_nth(n, c, s)` that returns the index of the nth occurrence of the character `c` in the string `s`, or `-1` if there is no such occurrence. For example:

- `index_nth(2, 'a', 'banana')` should return `3` because the 2nd occurrence of `'a'` in `'banana'` has an index of 3.
- `index_nth(3, 'a', 'banana')` should return `5` because the 3rd occurrence of `'a'` in `'banana'` has an index of 5.
- `index_nth(4, 'a', 'banana')` should return `-1` because there are not 4 occurrences of `'a'` in `'banana'`.

You must use a loop! You may ***not*** use any of the built-in string *methods*, although you may of course use *functions* like `range()` and `len()` as needed.

```
def index_nth(n, c, s):
    count = 0
    for x in range(len(s)):
        if s[x] == c:
            count += 1
        if count == n:
            return x

    if count != n:
        return -1
```

email
Jh00@bn.edu

Table of Hmmm Instructions

Instruction	Description	Aliases
System instructions		
Halt	Stop!	
read rX	Place user input in register rX	
write rX	Print contents of register rX	
Nop	Do nothing	
Setting register data		
setn rX N	Set register rX equal to the integer N (-128 to +127)	
addn rX N	Add integer N (-128 to 127) to register rX	
copy rX rY	Set rX = rY	**mov**
Arithmetic		
add rX rY rZ	Set rX = rY + rZ	
sub rX rY rZ	Set rX = rY - rZ	
neg rX rY	Set rX = -rY	
mul rX rY rZ	Set rX = rY * rZ	
div rX rY rZ	Set rX = rY / rZ (integer division; no remainder)	
mod rX rY rZ	Set rX = rY % rZ (returns the remainder of integer division)	
Jumps!		
jumpn N	Jump to line N	
jumpr rX	Jump to line number stored in rX	**jump**
jeqzn rX N	If rX == 0, then jump to line N	**jeqz**
jnezn rX N	If rX != 0, then jump to line N	**jnez**
jgtzn rX N	If rX > 0, then jump to line N	**jgtz**
jltzn rX N	If rX < 0, then jump to line N	**jltz**
calln rX N	Copy the next line number into rX and then jump to line N	**call**
Interacting with memory (RAM)		
loadn rX N	Load register rX with the contents of memory address N	
storen rX N	Store contents of register rX into memory address N	
loadr rX rY	Load register rX with data from the address location held in reg. rY	**loadi**, **load**
storer rX rY	Store contents of register rX into memory address held in reg. rY	**storei**, **store**

Blank page for extra work; do not detach!