IC210: Introduction to Computing

Fall AY2016 — 6-Week Exam

Individual work. Closed book. Closed notes.
You may not use any electronic device.
This is a multi section exam that will be given to different midshipmen at different times. As per USNAINST 1531.53A, you may NOT communicate about this exam with anyone using any medium until your instructor tells you that you can.

Name: ___________________________________, Alpha: ___________, Section Number: ___________

Instructor name: _____________________________________

ASCII Table for Printable Characters																				
Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
32	20		46	2e	.	60	3c	<	74	4a	J	88	58	X	102	66	f	116	74	t
33	21	!	47	2f	/	61	3d	=	75	4b	K	89	59	Y	103	67	g	117	75	u
34	22	"	48	30	0	62	3e	>	76	4c	L	90	5a	Z	104	68	h	118	76	v
35	23	#	49	31	1	63	3f	?	77	4d	M	91	5b	[	105	69	i	119	77	w
36	24	$	50	32	2	64	40	@	78	4e	N	92	5c	\	106	6a	j	120	78	x
37	25	%	51	33	3	65	41	A	79	4f	O	93	5d	]	107	6b	k	121	79	y
38	26	&	52	34	4	66	42	B	80	50	P	94	5e	^	108	6c	l	122	7a	z
39	27	'	53	35	5	67	43	C	81	51	Q	95	5f	_	109	6d	m	123	7b	{
40	28	(	54	36	6	68	44	D	82	52	R	96	60	`	110	6e	n	124	7c	|
41	29	)	55	37	7	69	45	E	83	53	S	97	61	a	111	6f	o	125	7d	}
42	2a	*	56	38	8	70	46	F	84	54	T	98	62	b	112	70	p	126	7e	~
43	2b	+	57	39	9	71	47	G	85	55	U	99	63	c	113	71	q			
44	2c	,	58	3a	:	72	48	H	86	56	V	100	64	d	114	72	r			
45	2d	-	59	3b	;	73	49	I	87	57	W	101	65	e	115	73	s			

Operator Name	**Associativity**	**Operators**
Primary scope resolution	left to right	::
Primary	left to right	() [ ] . -> dynamic_cast typeid
Unary	right to left	++ -- + - ! ~ & * (*type_name*) sizeof new delete
C Pointer to Member	left to right	.*->*
Multiplicative	left to right	* / %
Additive	left to right	+ -
Bitwise Shift	left to right	<< >>
Relational	left to right	< > <= >=
Equality	left to right	== !=
Bitwise AND	left to right	&
Bitwise Exclusive OR	left to right	^
Bitwise Inclusive OR	left to right	|
Logical AND	left to right	&&
Logical OR	left to right	||
Conditional	right to left	? :
Assignment	right to left	= += -= *= /= <<= >>= %= &= ^= |=
Comma	left to right	,

1. [9pts] Suppose you (successfully) give the command: **gcc prog1.c -o george**

   a. Which of these is an executable program (circle all that apply): **gcc** **prog1.c** **george**

   b. Which of these is a file (circle all that apply): **gcc** **prog1.c** **george**

   c. Which of these is created by the command (circle all that apply): **gcc** **prog1.c** **george**

2. [20pts] Assuming the following definitions, fill in the table. **Note:** each expression should be taken as independent. I.e. if one expression modifies some variable values, those modifications do not carry over to the next expression.

```
int b = 1;
int n = 0;
int j = 6;
double z = 1.5;
char c = 67;
stream fin = fopen("foo", "r");
    // assume foo exists in
    // the current directory
```

expression	type	value
c == 'c'		
c = 10*j + z		
(int)('3' * 2)		
b = n		
3/j + z		
j * z		
!fin		
j % 3 && b		
1 + j/10 > z		
n != 0 || n != 10		

3. [8pts] Programs are often given version numbers in the form **v***majorVersion***.***minorVersion* . A higher majorVersion number means a more recent version of the program. When the majorVersion numbers are the same, the larger minorVersion number denotes the more recent program. So, v3.2 is more recent than v1.8, and v3.12 is more recent than v3.9 (because 12 is bigger than 9).

   Fill in the below to create a program that reads in a version number (e.g. **v5.24**) and writes the word "newer" if the version is more recent than v2.7 and "not newer" otherwise. Here are some example runs:

~/$ ./go v1.8 not newer	~/$ ./go v2.7 not newer	~/$ ./go v2.15 newer	~/$ ./go v3.0 newer

```
#include "si204.h"
int main() {








  return 0;
}
```

4. [8pts] The code below reads in values i, j and k, then prints out the value of $i \times j^k$. Write a chunk of code that is equivalent to the code below, except that a while loop is used instead of a for loop.

```
int i, j, k, e=1;
i = readnum(stdin);
j = readnum(stdin);
k = readnum(stdin);

for(int i = 0; i < k; i++) {
  e = e*j;
}

fputs("result: ", stdout);
writenum(i*e, stdout);
fputs("\n", stdout);
```

5. [8pts] Consider sequences like this

d23 d2 u101 d5 u6 d19 x

representing a sequence of down (d) and up (u) moves. The x just terminates the sequence. Fill in the blank in the following code to make a program that reads a sequence like the above from the user (assume valid input, i.e. don't worry about error checking!) and prints out the sum of all the "down" moves, followed by the sum of all the "up" moves. A sample run would look like this

```
u9 d2 d7 u3 d1 x    ← user input
down = 10
up = 12
```

```
#include "si204.h"

int main() {
  int up = 0, down = 0;
```

```
  printf("down = %i\n", down);
  printf("up = %i\n", up);
  return 0;
}
```

6. [9pts] Fill in the conditions on the if statements on the right so that they are equivalent to the code on the left

a.

```
if (y == 0 || x == 0)                        if (____________________________________)
{                                            {
  // do nothing, avoids divide by zero         writenum(1/x + 1/y, stdout);
}                                            }
else
{
  writenum(1/x + 1/y, stdout);
}
```

b.

```
if (w % 2 == 0)                              if (____________________________________)
{                                            {
  if (d == 6)                                  b += p;
  {                                          }
    b += p;
  }
}
```

c.

```
if (a <= 'k')                                if (____________________________________)
{                                            {
  printf("easy\n");                            printf("easy\n");
}                                            }
else                                         else
{                                            {
  if (a >= 'q') {                              printf("tough\n");
    printf("easy\n");                        }
  } else {
    printf("tough\n");
  }
}
```

7. [8pts] For the following questions you might want to consult the operator precedence/associativity table on the front of this exam.

a. Is x = y != z evaluated as **(x = y) != z** or as **x = (y != z)** ? (circle one)
Explain your answer!

b. Is x == y != z evaluated as **(x == y) != z** or as **x == (y != z)** ? (circle one)
Explain your answer!

8. [10pts] The following program is supposed to read in a starting number from the user (we can assume a positive number) and countdown by two's from the starting number. It sometimes works, for example if you input 10 it outputs

```
10 8 6 4 2 blast off!
```

which is what I want. But it sometimes seems to get stuck in an infinite loop. For example, if you input 11 you get an infinite loop rather than what I was hoping for:

```
11 9 7 5 3 1 blast off!
```

a. Annotate the code to show how to fix this problem.

```
#include "si204.h"

int main() {
  fputs("Enter start: ", stdout);
  int k = readnum(stdin);

  for(int i = k; k != 0; k = k - 2) {
    writenum(k, stdout);
    fputs(" ", stdout);
  }

  fputs("blast off!\n", stdout);

  return 0;
}
```

b. Explain why the original fails sometimes, but not always.

9. [10pts] Give the type for each of the expressions identified below.

```
#include "si204.h"

int main()
{
  double bal = 700, pay = 550;
  double rate = 3.1, expense = 45;
  int week;
                _a_
               /   \
  week = readnum(stdin);
                  _____b_______
                 /             \
  for(int day = 1; day <= week * 7; day++)
  {      __c___
        /      \
    if (day % 14 == 13)
    {
      bal += pay;
    }
    bal -= expense;

    bal += bal*rate/1200;
  }            \_______/
                   d

  cstring m = "You are ";
  if (bal < 0) {
    strcat(m, "bankrupt");
  } else {
    strcat(m, "solvent");
  }           \_______/
                  e

  fputs(m, stdout);
  fputs("\n", stdout);

  return 0;
}
```

a. ______________________

b. ______________________

c. ______________________

d. ______________________

e. ______________________

10. [10pts] When I try to compile the code below, I get the following error messages:

```
p10.c: In function `main`:
p10.c:10:5: warning: format `%i` expects argument of type `int *`, but argument 3 has type `int`
p10.c:11:17: error: lvalue required as left operand of assignment
p10.c:20:5: error: `total` undeclared (first use in this function)
```

Annotate the code to show how to fix these errors. When you're done you should have a program that correctly either prints out -1, if file tmp.txt does not exist, or prints the sum of all the even numbers in tmp.txt, assuming it's just a file full of integers.

```
 1. #include <stdio.h>
 2.
 3. int main() {
 4.   FILE* fin = fopen("tmp.txt", "r");
 5.
 6.   if (fin != 0) {
 7.     int total = 0;
 8.     int x;
 9.
10.     while (fscanf(fin, " %i", x) == 1) {
11.       if (x % 2 = 0) {
12.         total = total + x;
13.       }
14.     }
15.
16.     fclose(fin);
17.   }
18.   else
19.   {
20.     total = -1;
21.   }
22.
23.   fprintf(stdout, "%i\n", total);
24.
25.   return 0;
26. }
```