IC210: Introduction to Computing

Fall AY2016 — 12-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 timesAs per USNAINST 1531.53A, you may NOT communicate about this exam with anyone using any medium until you 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	,

NOTE: This exam has been modified from the original to use C instead of C++. It has not been modified otherwise. The coverage from IC210 at 12 weeks is NOT the same as the coverage for SI204, so some topics might be different!

1. [11pts] In the following code, clearly identify (e.g. circle and label) every

   1. function prototype
   2. function definition
   3. function call
   4. function argument
   5. function parameter

   Note: function fabs() is defined in math.h withprototype double fabs(double);

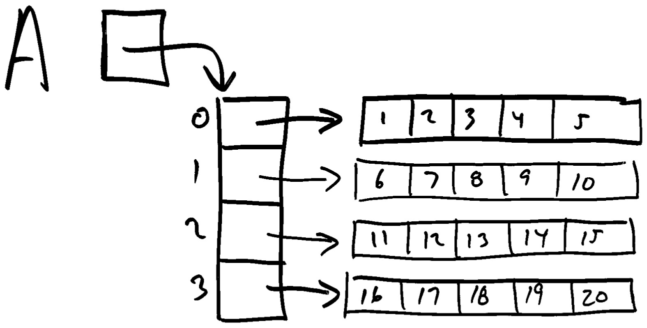
```
#include <math.h>
#include <stdio.h>
#include <string.h>

int rating2int(char* s);

double int2rating(int k);

int similar(int r1, int r2);

int main()
{
  int* R = calloc(20, sizeof(int));
  char N[20][128];
  char s[128];
  FILE* fin = fopen("bonds.txt", "r");
  for(int i = 0; i < 20; i++)
  {
    fscanf(fin, " %s %s", s, N[i]);
    R[i] = rating2int(s);
  }
  scanf(" %s", s);
  int t = rating2int(s);
  int k = 0;
  while(k < 20 && !similar(t,R[k]))
    k++;
  printf("%g %s\n", int2rating(R[k]), N[k]);
  return 0;
}

int rating2int(string s)
{
  return 3*(s[0] - 'A') + 3 - strlen(s);
}

double int2rating(int k)
{
  double res = 0.0;
  int c = k/3 + 'A';
  for(int i = 0; i < 3 - k % 3; i++)
    res += c;
  return res;
}

int similar(int r1, int r2)
{
  return fabs(r1-r2) <= 1;
}
```

2. [18pts] Write the type for each underlined expression.
   Note: function fabs() is defined in math.h withprototype double fabs(double);

```
#include <math.h>
#include <stdio.h>
#include <string.h>

int rating2int(string s);
double int2rating(int k);
int similar(int r1, int r2);

int main()
{
  int* R = calloc(20, sizeof(int));                ⇦ ⇦ ⇦ ⇦__________________
  char N[20][128];
  char s[128];
  FILE* fin = fopen("bonds.txt", "r");
  for(int i = 0; i < 20; i++)
  {
    fscanf(fin, " %s %s", s, N[i]);                ⇦ ⇦ ⇦ ⇦__________________
    R[i] = rating2int(s);                          ⇦ ⇦ ⇦ ⇦__________________
  }
  scanf(" %s", s);
  int t = rating2int(s);
  int k = 0;
  while(k < 20 && !similar(t,R[k]))                ⇦ ⇦ ⇦ ⇦__________________
    k++;
  printf("%g %s\n", int2rating(R[k]), N[k]);       ⇦ ⇦ ⇦ ⇦__________________
  return 0;
}

int rating2int(string s)
{
  return 3*(s[0] - 'A') + 3 - strlen(s);           ⇦ ⇦ ⇦ ⇦__________________
}

double int2rating(int k)
{
  double res = 0.0;
  int c = k/3 + 'A';                               ⇦ ⇦ ⇦ ⇦__________________
  for(int i = 0; i < 3 - k % 3; i++)
    res += c;
  return res;
}

int similar(int r1, int r2)
{
  return fabs(r1-r2) <= 1;                         ⇦ ⇦ ⇦ ⇦__________________
}
```

3. [12pts]

   a. Write the code (as it would appear in main(), for example) that creates a variable A and allocates and initializes so that we have the situation depicted in the picture to the right.



   b. Write a complete definition of functionprintcol that could be called like this **printcol(A,4,5,1)** (assuming the above situation) to print the index 1 column ofthe array A. Thus, the call printcol(A,4,5,1) would produce the output shown below.**Note:** the function should work for any 2D array of intswith proper dimensions and column number.

```
2
7
12
17
```

4. [12pts] Below a program I'd like to have. Write down the *prototypes* (not definitions) you would need for each of the functions called.
Note: Not that it really matters for this problem, but this program should read strings from data.txt and store them in the array d, then reorder the strings from shortest to longest, then print out all the strings containing "hap" (from shortest to longest).

```
int main()
{
  FILE* fin = fopen("data.txt", "r");
  int n;
  char** d = read(fin, &n);
  reorder(d, n);
  int k = -1;
  while((k = indexOfNextContain(d, n, "hap", k)) < n) {
    printf("%s\n", d[k]);
  }
  return 0;
}
```

a. Prototype for read:

b. Prototype for reorder:

c. Prototype for indexOfNextContain:

5. [5pts] What problem would arise if a program were to call the function **foo** defined below over and over and over again? **Note:** What value this function computes is not really relevant to answering this problem.

```
double foo(double x, double d, int n)
{
  double* D = calloc(n+1, sizeof(double));
  for(int i = 0; i <= n; i++) {
    D[i] = x + i*d;
  }
  while(n-- > 1) {
    for(int i = 0; i < n; i++) {
      D[i] = D[i+1] - D[i];
    }
  }
  return D[0];
}
```

6. [10pts]

   a. When I run the program below, no matter what the user types, it crashes:

```
~/$ ./ex03
100 90 80 70 60 50 40 30 20 10  ← user input in bold
Segmentation fault (core dumped)
```

   Annotate the code to show how to fix main() so that the program at least gets as far as printing out the message "Made it here!".

```
#include <stdio.h>

void specialPrint(int* A, int N, char c);

int main()
{
  char c = ',';
  int* A;
  int N = 10;
  for(int i = 0; i < N; i++) {
    scanf(" %i", &A[i]);
  }
  printf("Made it here!\n");
  specialPrint(A, N, c);
  return 0;
}

void specialPrint(int* A, int N, char c)
{
  for(int i = N; i > 1; i--) {
    printf("%i %c", A[i], c);
  }
  printf("%i\n", A[0]);
}
```

   b. When the Part a bug is fixed, the program no longer crashes, but it doesn't do what it's supposed to do, which is print the inputted 10 numbers in reverse order separated by commas. Instead here's what we get:

```
~/$ ./ex03
100 90 80 70 60 50 40 30 20 10
Made it here!
135121,10,20,30,40,50,60,70,80,100
```

   Show how to fix specialPrint!

7. [9pts]
NOTE for SI204: This is still in C++, but you get the idea of the kind of problem that you should expect where you see compiler messages and have to debug it.
When I try to compile the code below, I get the following error messages:

```
ex.cpp:39:26: error: invalid initialization of reference of type 'char&' from expression of type 'std::string'
ex.cpp:14:30: error: cannot convert 'std::string**' to 'std::string*' in initialization
ex.cpp:18:6: error:   initializing argument 3 of 'void show(std::string*, int, std::ostream)'
```

Annotate the code to show how to fix these errors.

```
 2 #include <stdio.h>
 3 #include <string>
 5
 6 void show(string *A, int n, ostream out);
 7 void modify(char& c, char bad, char cover);
 8 void cross(string *A, int n, char bad);
 9
10 int main()
11 {
12   int num;
13   cin >> num;
14   string* B = new string*[num];
15   for(int i = 0; i < num; i++)
16     cin >> B[i];
17   cross(B,num,'o');
18   show(B,num,cout);
19   return 0;
20 }
21
22 void show(string *A, int n, ostream out)
23 {
24   for(int i = 0; i < n; i++)
25     out << A[i] << ' ';
26   out << endl;
27 }
28
29 void modify(char& c, char bad, char cover)
30 {
31   if (c == bad)
32     c = cover;
33 }
34
35 void cross(string *A, int n, char bad)
36 {
37   for(int r = 0; r < n; r++)
38     for(int i = 0; i < A[r].length(); i++)
39       modify(A[i],bad,'x');
40 }
```

8. [8pts] I'd like a function "convert" that converts its argument from a time in seconds to a time in minutes, and returns the leftover time (in seconds).For example, this code

```
  int t = 137;
  double left = convert(&t);
  printf("%i minutes and %g seconds\n", t, left);
```

... should print out "2 minutes and 17 seconds". Define the function "convert" that works this way

9. [15pts] Consider the following function prototypes and variable declarations (assume arrays get allocated and initialized elsewhere).

```
int revi(int k); // returns the reverse of k, e.g. if k is 387, return 783
char* revs(char* s); // returns the reverse of s
void capit(char &c); // changes c to a capital letter if it's lower case
int iscap(char c); // returns true if c is a capital letter, false otherwise

char c = 'a';
char w[128] = "hello";
char* A;           \
char** B;          |-- assume arrays get allocated and initialized elsewhere!
int** C;           /
```

Fill in the table below with the types of the following expressions, or "error" if the expression i invalid.

expression	type
A	
A[i][0]	
C[0]	
C[i][i+1]	
revs("four")	
revi(4)	
3.5 + revi(12)	
capit(revs(B[0]))[1]	
revi(capit(w))	
capit(3.0)	