ECE264 Fall 2018 Final Exam

330-530 PM, December 10

Please keep the answer sheet clean. Do not use the answer sheet as your scratch space. The answer sheet should contain **only** the answers.

Do not write anything that is not the answers. Otherwise, you may lose points.

Please use **DARK** ink. If your pen is too light, your answer may not be graded.

The ASCII Table

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
00	00	NUL	32	20	SP	64	40	@	96	60	‘
01	01	SOH	33	21	!	65	41	A	97	61	a
02	02	STX	34	22	"	66	42	B	98	62	b
03	03	ETX	35	23	#	67	43	C	99	63	c
04	04	EOT	36	24	$	68	44	D	100	64	d
05	05	ENQ	37	25	%	69	45	E	101	65	e
06	06	ACK	38	26	&	70	46	F	102	66	f
07	07	BEL	39	27	’	71	47	G	103	67	g
08	08	BS	40	28	(	72	48	H	104	68	h
09	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

1 Huffman Tree

Consider the beginning of the **header** of a Huffman compressed file is (output of `xxd`)

```
bc5e 57a5 86c5 6300  ...
```

Here ... means more to come (for the rest of the header). For your reference, the output of `xxd -b` for the same file is shown below (empty line is added for clarity).

```
10111100 01011110 01010111 10100101

10000110 11000101 01100011 00000000 ...
```

When the control bit is `1`, it is followed by 8 bits of data as a letter.

A: Which statement is correct?

1. The code for `p` is 101 (binary).
2. The code for `x` is 000 (binary).
3. The code for `w` is 010 (binary).
4. The code for `a` is 11 (binary).
5. The code for `u` is 111 (binary).
6. The code for `t` is 100 (binary).
7. None of the above

B: Which statement is correct?

1. The count of `b` must be larger than the count of `a`.
2. The count of `x` must be larger than the count of `z`.
3. The count of `p` must be larger than the count of `x`.
4. The count of `c` must **not** be greater than the count of `u`.
5. The count of `b` must **not** be greater than the count of `a`.
6. The count of `a` must **not** be greater than the count of `x`.
7. None of the above

C: Which statement is correct?

1. The code for `a` has the same length as the code for `c`.
2. The code for `a` has the same length as the code for `x`.
3. The code for `p` has the same length as the code for `u`.
4. The code for `a` has the same length as the code for `z`.
5. The code for `b` has the same length as the code for `w`.
6. The code for `x` has the same length as the code for `z`.
7. None of the above

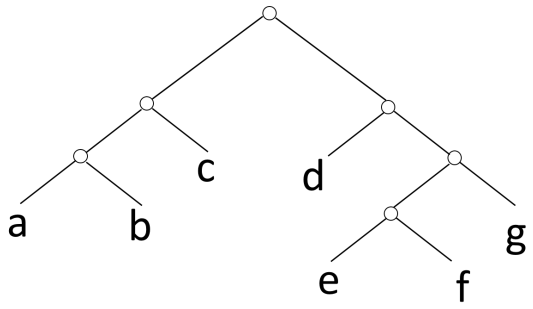
D: Which statement is correct?

1. Four different letters appear in the original file.
2. Five different letters appear in the original file.
3. Six different letters appear in the original file.
4. Seven different letters appear in the original file.
5. At least eight different letters appear in the original file.
6. It is not possible to use Huffman compression if n different letters appear in the original file and n is an odd number.
7. It is not possible to use Huffman compression if each letter needs six bits, instead of eight.
8. None of the above

Answer:
2 5 4 3

2 Huffman Compression

Consider the following tree expressing the codes for Huffman compression.



Consider the beginning of the **data** of a Huffman compressed file is (output of `xxd`)

```
bc5e 57a5 86c5 6300  ...
```

Here `...` means more to come. For your reference, the output of `xxd -b` for the same file is shown below (empty line is added for clarity).

```
10111100 01011110 01010111 10100101

10000110 11000101 01100011 00000000 ...
```

Write down the **first**, **third**, **fifth**, and **seventh** letters of the data for answers A - D.
Answer:

```
10: d
111: g
10: d
001: b
01: c
111: g
001: b
01: c
01: c
111: g
0100101 10000110 11000101 01100011 00000000

d d c b
```

3 Shape of Binary Trees

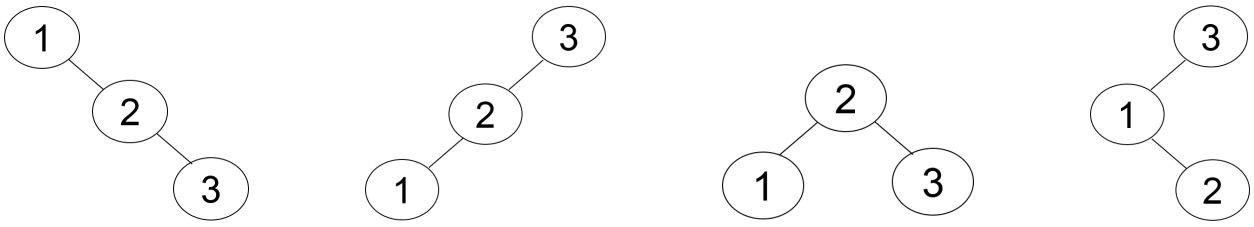
Define $NL(n)$ as the number of nodes on the left side of node n; $NR(n)$ is the number of nodes on the right side of node n. Two trees T_1 and T_2 have same shape if

- $NL(r_1)$ is the same as $NL(r_2)$; $NR(r_1)$ is the same as $NR(r_2)$. Here, r_1 and r_2 are the roots of T_1 and T_2 respectively.
- The left side of r_1 has the same shape as the left side of r_2.
- The right side of r_1 has the same shape as the right side of r_2.

Please notice that this definition is recursive.

The shape of a binary search tree depends on the order of data.

- If 1, 2, 3 are inserted in the order 1, 2, 3, the binary search tree is shown in Figure 1 (a).
- If the order is 3, 2, 1, the binary search tree is shown in Figure 1 (b).
- If the order is 2, 1, 3, the binary search tree is shown in Figure 1 (c). Please notice that if the order is 2, 3, 1, the tree is the same.
- If the order is 3, 1, 2, the binary search tree is shown in Figure 1 (d).



(a) (b) (c) (d)

Figure 1: Different shapes of binary search trees.

A: There are 24 (4!) different orders of 1, 2, 3, 4. How many different orders can produce the same shape as the sequence 2, 1, 4, 3 (including 2, 1, 4, 3 itself)?

B: There are 120 (5!) different orders of 1, 2, 3, 4, 5. How many different orders can produce the same shape as the sequence 4, 2, 3, 1, 5 (including 4, 2, 3, 1, 5 itself)?

C: A binary tree (may not be a binary search tree) has 6 nodes. It has the same output for in-order and pre-order traversal. Each node stores distinct data. What is the **maximum** number of different **shapes** this tree may have? Hint: Your answer may be 0, 1, 2, 3, ... infinity.

D: A binary tree (it may not be a binary search tree) has the same output for in-order, pre-order, and post-order traversal. What is the **maximum** number of **nodes** in this tree? If the tree has more than one nodes, each node stores distinct data. Hint: Your answer may be 0, 1, 2, 3, ..., infinity.

Answer:
3: 2 1 4 3, 2 4 3 1, 2 4 3 1
8: 42315, 42351, 42135, 42153, 42513, 42531, 45213, 45231
1: the nodes have only right children
1

4 Reverse Linked List

This program asks you to write a function that reverses a linked list by reversing the individual links between the nodes. The function's input argument is the head of the linked list and returns the head of the reversed linked list. This function should not call `malloc`. Figure 2 shows an example list and its reversed form.

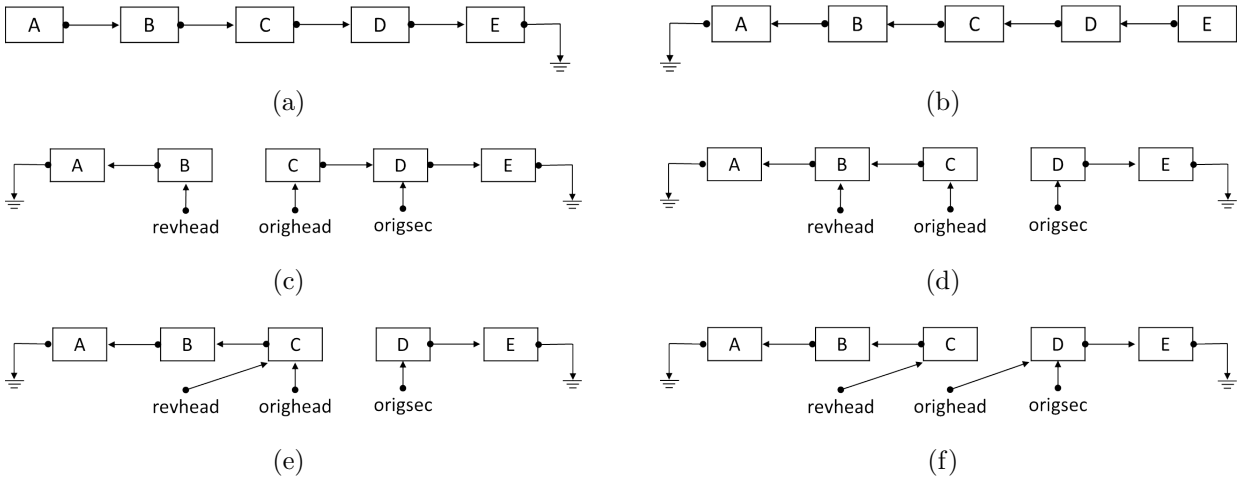


Figure 2: (a) The original linked list. The list's head points to `A`. (b) The reversed linked list. The list's head points to `E`. The steps from (a) to (b) are shown in (c)-(f).

Consider the following program with one (or several) mistake(s). A linked list from Figure 2 (a) is given (`A` is the head node).

```
1  #include "list.h"
2  Node * List_reverse(Node * head)
3  {
4    if (head == NULL)
5      {
6        // empty list, nothing to do
7        return NULL;
8      }
9
10   // Answer C, consider the following three lines of code
11   Node * orighead = head; // C.1
12   Node * revhead = NULL;  // C.2
13   Node * origsec;         // C.3
14   while (orighead != NULL)
15     {
16       // The following code has one (or several) mistake(s)
17       // DO NOT CORRECT THE MISTAKE
18       origsec = orighead -> next;
19       orighead -> next = revhead;
20       orighead = origsec;
21       revhead = orighead;
22       // <--- Answer D
23     }
24   return revhead;
25 }
```

A: Which node is the new head returned by the function?

1. `A`
2. `B`
3. `C`
4. `D`
5. `E`
6. Segmentation fault
7. Infinite loop
8. May be different when the program runs again, but the program has no segmentation fault and no infinite loop
9. None of the above

B: How many nodes' memory is lost? Your answer may be 0 (no memory leak), 1, 2, 3, 4, or 5 (all nodes' memory is lost).

C: Can you correct the mistake(s) by removing or reordering the three lines marked as `C.1` `C.2`, and `C.3`? If the answer is no (reordering or removing these three lines cannot correct the code), write No. If the answer is yes, write the order that can correct the code. If the answer is yes, your answer may be something like
`C.2, C.1, C.3` (meaning reordering `C.2` and `C.1`)
or
`C.1, C.3` (meaning that `C.2` should be deleted).

D: Can you add code (location marked for `Answer D`) to correct the mistake? You can **only add** code. You are not allowed to delete anything within `while`. You are not allowed to recorder anything within `while`. You **cannot change anywhere else** (other than the correction for Answer C, if you have any). If the answer is no (nothing at this location can correrct the code), write No. If the answer is yes, write the code that can correct the error.
Answer:
9
5
No.
No

5 Integer Partition

A positive integer can be expressed as the sum of a sequence of positive integers, or itself. *Integer partition* creates such sequences of integers. For example, 5 can be broken in to the sum of 1 + 2 + 2 or 2 + 3. These two partitions use different numbers, and thus are considered unique partitions. The order of the number in the partition is also important. Thus, 1 + 2 + 2 and 2 + 1 + 2 are considered different partitions because 1 appears in different positions. Below are some examples of integer partitions:

```
1 = 1          2  = 1 + 1        3  = 1 + 1 + 1     4  = 1 + 1+ 1 + 1
                  = 2               = 1 + 2            = 1 + 1 + 2
                                    = 2 + 1            = 1 + 2 + 1
                                    = 3                = 1 + 3
                                                       = 2 + 1 + 1
                                                       = 2 + 2
                                                       = 3 + 1
                                                       = 4
```

In general, number n can be partitioned in 2^{n-1} different ways. For example, 4 can be partitioned in $2^{4-1} = 8$ different ways.
If 1 cannot be used, 2 can be partitioned in only one way: 2.
If 1 cannot be used, 4 can be partitioned in only two ways: 2 + 2 and 4.

A: If 1 cannot be used, how many ways can 6 be partitioned?

B: If 1 cannot be used, how many ways can 10 be partitioned?

Consider the following program with one mistake.

```
 1 // partition.c
 2 #include <stdio.h>
 3 #include <stdlib.h>
 4 #include <string.h>
 5 void printPartition(int * arr, int length)
 6 {
 7   int ind;
 8   for (ind = 0; ind < length - 1; ind ++)
 9     {
10       printf("%d + ", arr[ind]);
11     }
12   printf("%d\n", arr[length - 1]);
13 }
14
15 void partition(int * arr, int ind, int left)
```

```
16 {
17   int val;
18   if (left == 0)
19     {
20       printPartition(arr, ind);
21       return; // not necessary
22     }
23   for (val = 1; val < left; val ++)
24     // <---
25     // ERROR, should be <=, but it is < here
26     // DO NOT CORRECT THE MISTAKE
27     // --->
28     {
29       arr[ind] = val;
30       partition(arr, ind + 1, left - val);
31     }
32 }
33
34 int main(int argc, char * argv[])
35 {
36   if (argc != 2)
37     {
38       return EXIT_FAILURE;
39     }
40   int n = (int) strtol(argv[1], NULL, 10);
41   if (n <= 0)
42     {
43       return EXIT_FAILURE;
44     }
45   int * arr;
46   arr = malloc(sizeof(int) * n);
47   partition(arr, 0, n);
48   free (arr);
49   return EXIT_SUCCESS;
50 }
```

How many lines does this program print when the input number

C: `n` in `main` is 6.

D: `n` in `main` is 10.

Answer:

5

```
34

0 for both
```