Name: _____ Login: _____ CLOSED BOOK

Quiz 2 – October 5, 2016 – ECE 264 Fall 2016

The following code compiles correctly but may have one or more bugs.

```
 1 #include <stdio.h>
 2 #include <stdlib.h>
 3 struct Node {
 4     int    value;
 5     struct Node* prev;
 6     struct Node* next;
 7 };
 8 int main(int argc, char *argv[]) {
 9     struct Node* hop = malloc(sizeof(*hop));
10     hop -> value     = 1;
11     hop -> prev      = NULL;
12     hop -> next      = malloc(sizeof(*(hop -> next)));
13     struct Node* oat = hop -> next;
14     oat -> value     = 9;
15     oat -> prev      = hop;
16     oat -> next      = hop;
17     struct Node* yam = hop -> next -> prev;
18     printf("yam: %d\n", yam -> value);
19     for(struct Node* curr = hop; curr != NULL; curr = curr -> next) {
20         printf("%d", curr -> value);
21     }
22     free(hop -> next);
23     free(hop);
24     return EXIT_SUCCESS;
25 }
```

Fill in the blanks.

1. Line 18 prints [yam: 1].
2. The rest of the program (lines 19 to 25) prints [1919191919191919...].
3. Write the stack and heap contents as of just before line 18 is executed. Include the type, name, and value of all four local variables (including curr); the value of any heap memory (broken out by fields); a lock to mark any allocation blocks; and the addresses of all data, including the address of the next blank slot (so we can see the size of everything). For any uninitialized memory, write "garbage". There is an example on the other side of this sheet.

Stack

addr.*	type*	name*	value	part	fn
200	int	argc	1	arguments	main
204	char**	argv	→ {"./gum"}		
212	void*			return addr.	
220	struct Node*	hop	400	local vars	
228	struct Node*	oat	420		
236	struct Node*	yam	400		
244	struct Node*	curr	garbage		
252					

Heap

addr.*	value	
400	.value=1	
	.prev=NULL	
	.next=420	
420	.value=9	
	.prev=400	
	.next=400	
440		

Assume `sizeof(int)==4`, `sizeof(void*)==8`, and the program is called "gum".

This version corrects a small typo in #3 (`main()` → `bike()`)

Example

This example is just to clarify any questions about notation.

You do not need to write anything on this side of the quiz.

```
1 #include <stdio.h>
 2 #include <stdlib.h>
 3 struct Point {
 4     int x;
 5     int y;
 6 };
 7 int main(int argc, char *argv[]) {
 8     struct Point* p1 = malloc(sizeof(*p1))
 9     p1 -> x = 10;
10     p1 -> y = 11;
11     struct Point* p2 = malloc(sizeof(*p2))
12     p2 -> x = 12;
13     p2 -> y = 13;
14     free(p1);
15     free(p2);
16     return 0;
17 }
```

The following shows the stack and heap contents as of just before line 14.

Stack

addr.*	type*	name*	value	part	fn
200	int	argc	1	arguments	main
204	char**	argv	→ {"./gum"}		
212	void*			return addr.	
220	struct Point*	p1	400	local vars	
228	struct Point*	p2	408		
236					

Heap

addr.*	value	
400	.x = 10	
	.y = 11	
408	.x = 12	
	.y = 13	
416		

Assume `sizeof(int)==4`, `sizeof(void*)==8`, and the program is called "gum".

You do not need to write anything on this side of the quiz.

This version corrects a small typo in #3 (`main()` → `bike()`)