

CS241 SP15 Exam 3: Solution Key

Name: Geng, Y.

UIN: 656934543

Exam code: CCBADB

NetID: yangeng2

SCROLL TO THE NEXT PAGE TO REVIEW YOUR ANSWERS

A VERSION OF THESE QUESTIONS MAY APPEAR IN A FUTURE QUIZ

For the next set of questions, suppose the heap is managed with a linked list. Each node in the list is either allocated or free. The list is sorted by address. When `malloc()` is called, the list is searched for a free segment that is big enough, that segment is divided into an allocated segment (at the beginning) and a free segment. When `free()` is called, the corresponding segment should merge with its neighboring segments, if they are also free. A process has a heap which is initially unallocated. During its execution, the process issues a sequence of memory allocate/de-allocate calls (assume `pA`, `pB` etc are `void*` pointers). In all cases, break ties by choosing the earliest segment. Also, assume the algorithm allocates memory from the beginning of the free segment.

Hint: A simple block-based notation e.g. 2 (1) 4 (3) ()=unallocated may be useful to track the allocated and unallocated blocks of heap memory.

1. (1 point.) Use the above description of a linked-list heap allocator with a *Worst-fit* placement for a 14 MB heap. Determine the largest hole after the following code completes.

```
pA=malloc(3MB)
pB=malloc(4MB)
pC=malloc(2MB)
free(pB)
pD=malloc(2MB)
free(pA)
pE=malloc(4MB)
```

- (A) 4MB
- (B) 6MB
- (C) 2MB
- (D) 3MB
- (E) 5MB

2. (1 point.) What allocation scheme is used in the following example? Use the above description of a linked-list heap allocator with a 16KB heap. The current allocation is shown below. `malloc(1KB)` returns a pointer to 11K.

Address	0K	1K	2K	3K	4K	5K	6K	7K	8K	9K	10K	11K	12K	13K	14K	15K
Allocated?	Y			Y	Y		Y			Y	Y					

- (A) Worst-fit
- (B) None of the other answers are correct
- (C) First-fit
- (D) Best-fit

3. (1 point.) Use the above description of a linked-list heap allocator with a *First-fit* placement for a 17 MB heap. What is the largest possible malloc request (`pF=malloc(?)`) that can succeed after the following code completes?

```
pA=malloc(6MB)
pB=malloc(5MB)
pC=malloc(1MB)
free(pB)
pD=malloc(1MB)
free(pA)
pE=malloc(4MB)
```

- (A) 8MB
- (B) 4MB
- (C) 5MB
- (D) 7MB
- (E) 6MB

4. (1 point.) Which one of the following does not correctly describe one disadvantage of different placement strategies when used with an explicit linked-list implementation?

- (A) Best-fit creates tiny unusable holes
- (B) Best-fit search is slower than first-fit search
- (C) First-fit requires linked list to be maintained in increasing size-order
- (D) Worst-fit reduces large holes making large allocations impossible without additional heap memory

5. (1 point.) Use the above description of a linked-list heap allocator with a *Best-fit* placement for a 20 MB heap. Determine the number of holes after the following code completes.

```
pA=malloc(5MB)
pB=malloc(4MB)
pC=malloc(2MB)
free(pB)
pD=malloc(2MB)
free(pA)
pE=malloc(2MB)
```

- (A) 0
- (B) 1
- (C) 3
- (D) 4
- (E) 2

6. (1 point.) Which response does NOT describe a Boundary Tags -based allocator described by Donald Knuth?
- (A) Is an implicit linked list implementation
 - (B) Requires a buddy allocator to coalesce adjacent blocks
 - (C) Traverse allocated blocks by using their size
 - (D) Coalesces blocks to prevent false-fragmentation
 - (E) Store size of block at the beginning and end of the block.

7. (1 point.) Which expression is the best choice to set the value of `result` to 37 ?

```
long* pA = calloc(sizeof(long),1);  
pA[0]= 37;  
long* pB = (long*) realloc(pA, sizeof(long)*2);  
long result = _____;
```

- (A) `*(pA + 0)`
- (B) `*pB`
- (C) None of other responses are correct;
- (D) `pA[0]`
- (E) `pA[1]`

8. (1 point.) Which of the following will NOT reserve enough memory for 6 character pointers? Assume a character pointer requires 8 bytes of storage

- (A) `malloc(sizeof(char*) * 8);`
- (B) `calloc(64,1);`
- (C) `calloc(8, sizeof(char*));`
- (D) `malloc(64);`
- (E) All of the other responses reserve sufficient memory

9. (1 point.) Which one of the following is true for pthreads?
- (A) Include `pthread.h` to get declarations for `pthread_create` etc
 - (B) Add "`-multithreaded.h`" gcc compiler option to build multi-threaded programs
 - (C) If a multi-threaded process is `fork`-ed then the child process is also multi-threaded
 - (D) None of the other responses are correct

10. (1 point.) Which of the following is NOT true?
- (A) Some C library functions e.g. `asctime`, `strtok` are not thread-safe
 - (B) A function that uses static (global) variable to hold a result value is not thread-safe
 - (C) If a function is documented as “not thread-safe” then it must not be used in multi-threaded programs
 - (D) Two threads can use the function at the same time if it is “thread-safe”

11. (1 point.) Which one of the following is NOT true for `calloc`?
- (A) Use `free` to release (de-allocate) memory reserved by `calloc`.
 - (B) Allocates memory in a character stack
 - (C) Returns NULL if memory allocation failed
 - (D) Memory allocated by `calloc` will be initialized to zero
 - (E) `calloc(4,4)` is identical to `calloc(1,16)`

12. (1 point.) Which of the following will NOT cause a multi-threaded process (with multiple threads currently running) to terminate?

- (A) A background thread calls `exit`
- (B) The original thread calls `pthread_exit` from `main`
- (C) The original thread `returns` from `main`
- (D) The process is delivered a `SIGKILL` signal
- (E) A background thread writes to address zero

13. (1 point.) When a linux heap allocator requires more heap memory it can call
- (A) `sbrk`
 - (B) No system call is required; heap space is allocated automatically by the MMU
 - (C) `malloc`
 - (D) `heap_alloc`
 - (E) **Ghostbusters**. Just kidding. Hint this response is incorrect.

14. (1 point.) Four students were asked to write four alternative ways to print CS and a newline to standard output. Carefully read the four functions below and for each one, decide if it will print CS without error. Choose the most accurate response below.

```
void F1() { char *s = "C"; strcat(s, "S"); write(1,s,sizeof(s)); }
void F2() { char s[100]; *s=0; strcat(s, "CS"); printf("%s\n", s); }
void F3() { char *s=(char*)calloc(1,100); strcat(s,"CS"); puts(s); free(s); }
void F4() { static char s[100]; sprintf(s,"C"); strcat(s,"S\n"); write(1,s,strlen(s)); }
```

- (A) 2 functions are correct
- (B) None of the functions are correct
- (C) All 4 functions are correct
- (D) Only 1 function is correct
- (E) 3 functions are correct

15. (1 point.) In a POSIX system (such as LINUX) which one of the following is INCORRECT or FALSE by default?
- (A) A multithreaded process can use more than one CPU at a time.
 - (B) Processes can not directly read another process's global variables.
 - (C) Parent processes can not write into the memory of a child process.
 - (D) Processes running for the same user, must be careful not to overwrite each other's heap memory.
 - (E) Child processes can not write into the memory of a parent process.

16. (1 point.) Using an initial heap size of 2^{10} bytes (1024 Bytes) and a binary buddy-allocator, how many memory allocation requests of 18 bytes can be completed before the allocator requires additional heap memory?

- (A) 31 - 33 (inclusive)
- (B) 30 or fewer
- (C) 36 - 55 (inclusive)
- (D) 34 - 35 (inclusive)
- (E) 56 or greater

17. (1 point.) My multi-threaded process has one heap, two open file descriptors, three stacks and is running on a four core machine. How many threads are in my multi-threaded process ?

- (A) 3
- (B) None of the other answers are correct
- (C) 4
- (D) 2
- (E) 1

18. (1 point.) Assuming `calloc` succeeds will the following program crash (seg fault)? If so, where?

```
1 int * ptrA = calloc(1, 16000);  
2 int ** ptrB = & ptrA;  
3 int *** ptrC = & ptrB;  
4 *ptrC= NULL;  
5 int result = **ptrB ;
```

- (A) The program will not crash
- (B) Line 5
- (C) Line 2
- (D) Line 4
- (E) Line 3

19. (1 point.) Parameter values and automatic (local) variables are stored?
- (A) In the stack
 - (B) In global storage
 - (C) Below the start of the heap
 - (D) In the heap
 - (E) In read-only memory

20. (1 point.) Which one of the following is NOT true for the Buddy allocator compared to other memory allocators?
- (A) Minimizes fragmentation
 - (B) Can be used as a heap allocator
 - (C) Optimizes for performance
 - (D) Uses a hierarchy of allocation blocks of size 2^n

21. (1 point.) Which line,if any, will likely crash the program?

```
1 int main() {  
2   int a = 10,**c, *d;  
3   c = &d;  
4   *c = &a;  
5   **c= 5;  
6   d = NULL;  
7   return 0;  
}
```

(A) 4

(B) 3

(C) 6

(D) The program will not crash

(E) 5

22. (1 point.) For a linked-list heap allocator, which response best describes the following statements about implicit free lists when compared to explicit free lists? For implicit free lists...

- 1 Find-first allocation algorithm can be mapped onto different placement strategies
- 2 Require separate storage outside of the heap for the implicit linked list
- 3 Require additional operating system support to manage the heap
- 4 Can increase allocation time (compared to explicit free lists)

- (A) None of the other responses are correct
- (B) Only 1 is correct
- (C) Only 2 and 3 are correct
- (D) Only 4 is correct
- (E) Only 1 and 4 are correct

23. (1 point.) What will the following code print?

```
1 void* func(void*p) {
2     printf((char*)p) ; return "DEF";
3 }
4 int main() {
5     pthread_t id;
6     pthread_create(&id,NULL, func,"123");
7     void *r = NULL;
8     pthread_exit(& r);
9     printf(r); return 0;
10 }
```

- (A) DEF and nothing else is possible
- (B) DEF123 and nothing else is possible
- (C) 123DEF and nothing else is possible
- (D) None of the other responses are correct
- (E) 123 and nothing else is possible

24. (1 point.) Which of the following is NOT true?

- (A) Creating threads is faster than forking process
- (B) pthreads in the same process share the same heap and the same virtual memory address space
- (C) pthreads are peers; there is no hierarchy of threads in the same process
- (D) `pthread_exit` waits until all other threads finish before returning
- (E) Without `pthread_join`, long running multi-threaded programs can suffer from thread-zombies causing future `pthread_create` calls to fail.

Summary of answers:

Question	Correct Answer	Your Answer	Points
1	D	D	1
2	A	A	1
3	C	C	1
4	C	C	1
5	E	E	1
6	B	B	1
7	B	B	1
8	E	E	1
9	A	A	1
10	C	C	1
11	B	B	1
12	B	B	1
13	A	A	1
14	E	E	1
15	D	D	1
16	A	A	1
17	A	A	1
18	B	B	1
19	A	A	1
20	A	A	1
21	D	D	1
22	D	D	1
23	E	C	0
24	D	D	1
Total			23