A. int * ra;

   int * rb;

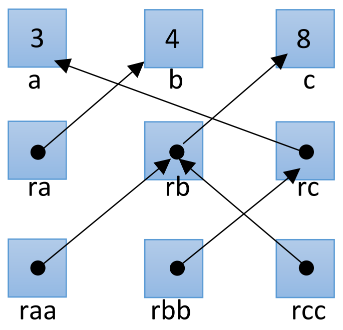
   int * rc;

   int ** raa;

   int ** rbb;

   int ** rcc;

B.



C. 3 4 8

   4 8

   8 3 8

D. ra = *b;

2.

A. Yes, it has a safety issue there. Since it is going to access “current_cell -> next” after we free the current_cell. In the heap, the memory will be treated as ‘freed memory’. Therefore, the program might change(overwrite) the memory which cause wrong result or seg. fault.

B.
```
for (current_cell=head; current_cell != NULL;){
    list *tmp = current_cell;
    current_cell = current_cell->next;
    free(tmp);
}
```

3. it is not safe to optimize (const folding).

ex)

declare new int pointer variable

int * p = NULL;

and then

a=4;

b=3;

p = &a;

***p = 7;**

…

<u>c=a-b; /* now c will be 7-3, which is 4 not 4-3 which is 1.*/</u>

4.

A. we cannot do dynamic allocation on the stack. Dynamic allocation defines amount of data that we need during runtime, but the stack only allows static allocation that defines data size at compile time.

B. we don’t have to but better to have because of the order of function procedures. For example, for the recursion, the program will be easy to track the steps in runtime stack.

C. yes, we could change our runtime environment to allocate all stack frames on the heap. The advantage of this is we are now having a single memory management for all. The disadvantage is the speed. Stacks are only valid in their subroutine but heap is not. we have to use heap operation to access the information. Also, we need to malloc and free the stack memory after the subroutine is done.