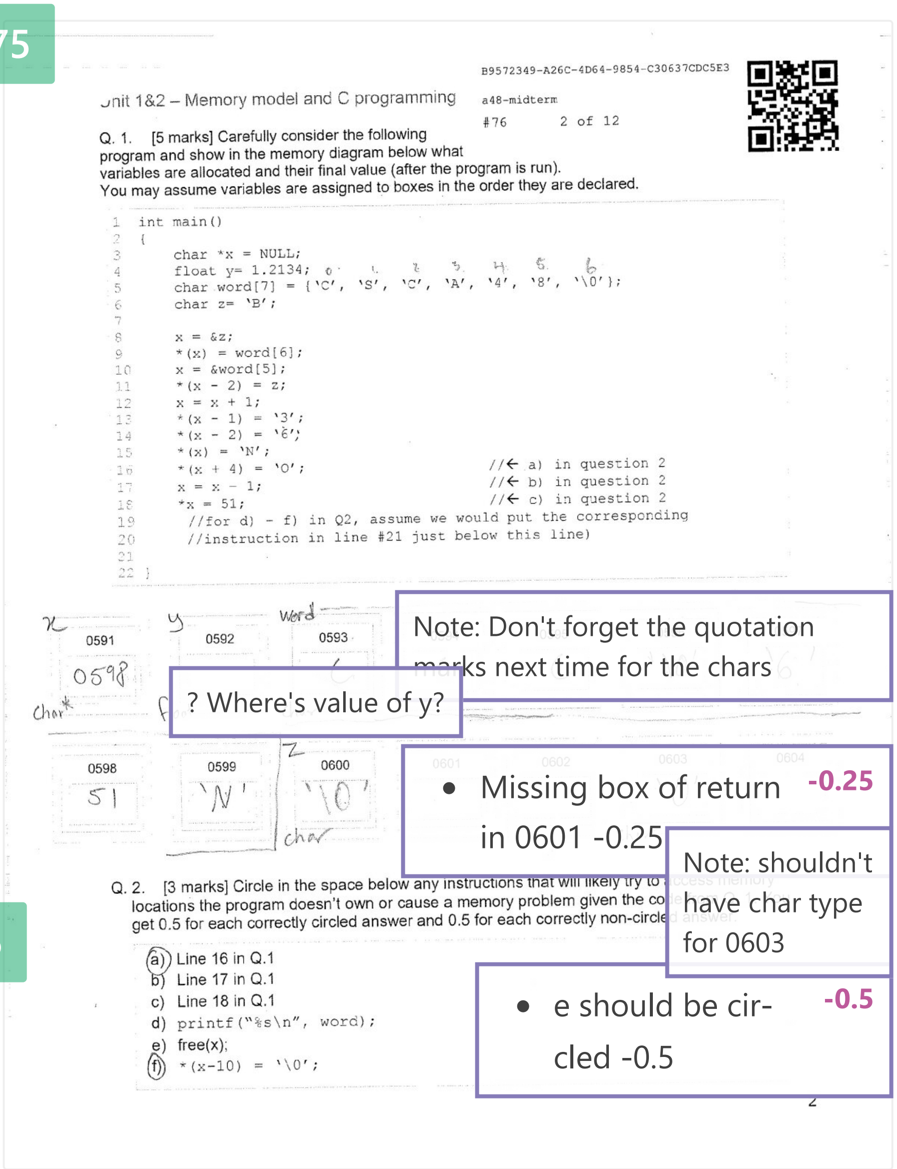
Q1 4.75



B9572349-A26C-4D64-9854-C30637CDC5E3

a48-midterm

#76 2 of 12

Unit 1&2 – Memory model and C programming

Q. 1. [5 marks] Carefully consider the following program and show in the memory diagram below what variables are allocated and their final value (after the program is run). You may assume variables are assigned to boxes in the order they are declared.

```
1   int main()
2   {
3       char *x = NULL;
4       float y= 1.2134;   0     1     2     3     4     5     6
5       char word[7] = {'C', 'S', 'C', 'A', '4', '8', '\0'};
6       char z= 'B';
7
8       x = &z;
9       *(x) = word[6];
10      x = &word[5];
11      *(x - 2) = z;
12      x = x + 1;
13      *(x - 1) = '3';
14      *(x - 2) = '6';
15      *(x) = 'N';
16      *(x + 4) = 'O';                    //← a) in question 2
17      x = x - 1;                         //← b) in question 2
18      *x = 51;                           //← c) in question 2
19       //for d) - f) in Q2, assume we would put the corresponding
20       //instruction in line #21 just below this line)
21
22  }
```

x
0591
0598
Char*

y
0592
f

word
0593
C

Note: Don't forget the quotation marks next time for the chars

? Where's value of y?

0598
51

0599
'N'

z
0600
'\0'
char

- Missing box of return in 0601 -0.25 **-0.25**

Note: shouldn't have char type for 0603

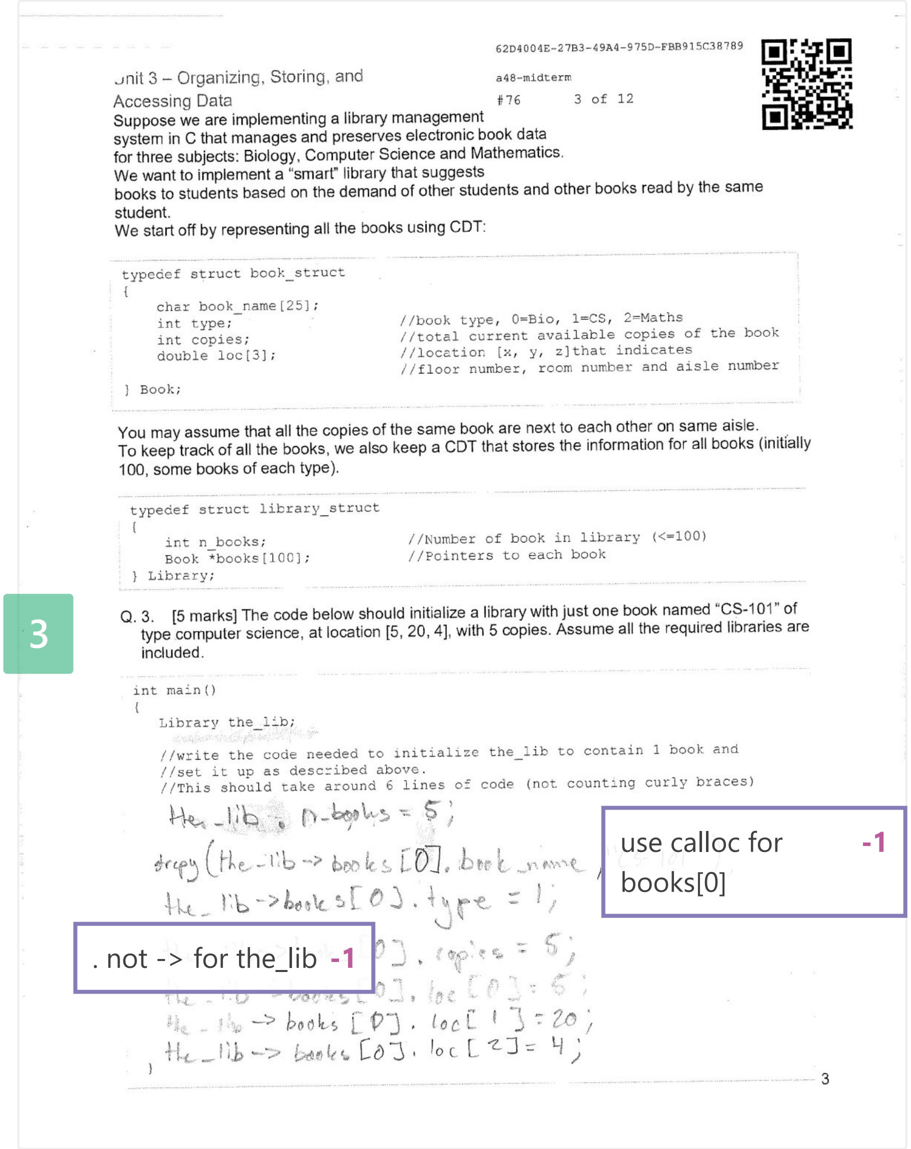
Q. 2. [3 marks] Circle in the space below any instructions that will likely try to access memory locations the program doesn't own or cause a memory problem given the code from Q. 1. You get 0.5 for each correctly circled answer and 0.5 for each correctly non-circled answer.

Q2 2.5

[✓ a)] Line 16 in Q.1
b) Line 17 in Q.1
c) Line 18 in Q.1
d) `printf("%s\n", word);`
e) free(x);
[✓ f)] `*(x-10) = '\0';`

- e should be circled -0.5 **-0.5**

2



62D4004E-27B3-49A4-975D-FBB915C38789
a48-midterm
#76 3 of 12

Unit 3 – Organizing, Storing, and Accessing Data

Suppose we are implementing a library management system in C that manages and preserves electronic book data for three subjects: Biology, Computer Science and Mathematics.
We want to implement a "smart" library that suggests books to students based on the demand of other students and other books read by the same student.
We start off by representing all the books using CDT:

```
typedef struct book_struct
{
    char book_name[25];
    int type;                   //book type, 0=Bio, 1=CS, 2=Maths
    int copies;                 //total current available copies of the book
    double loc[3];              //location [x, y, z]that indicates
                                //floor number, room number and aisle number
} Book;
```

You may assume that all the copies of the same book are next to each other on same aisle.
To keep track of all the books, we also keep a CDT that stores the information for all books (initially 100, some books of each type).

```
typedef struct library_struct
{
    int n_books;                //Number of book in library (<=100)
    Book *books[100];           //Pointers to each book
} Library;
```

Q3 3

Q. 3. [5 marks] The code below should initialize a library with just one book named "CS-101" of type computer science, at location [5, 20, 4], with 5 copies. Assume all the required libraries are included.

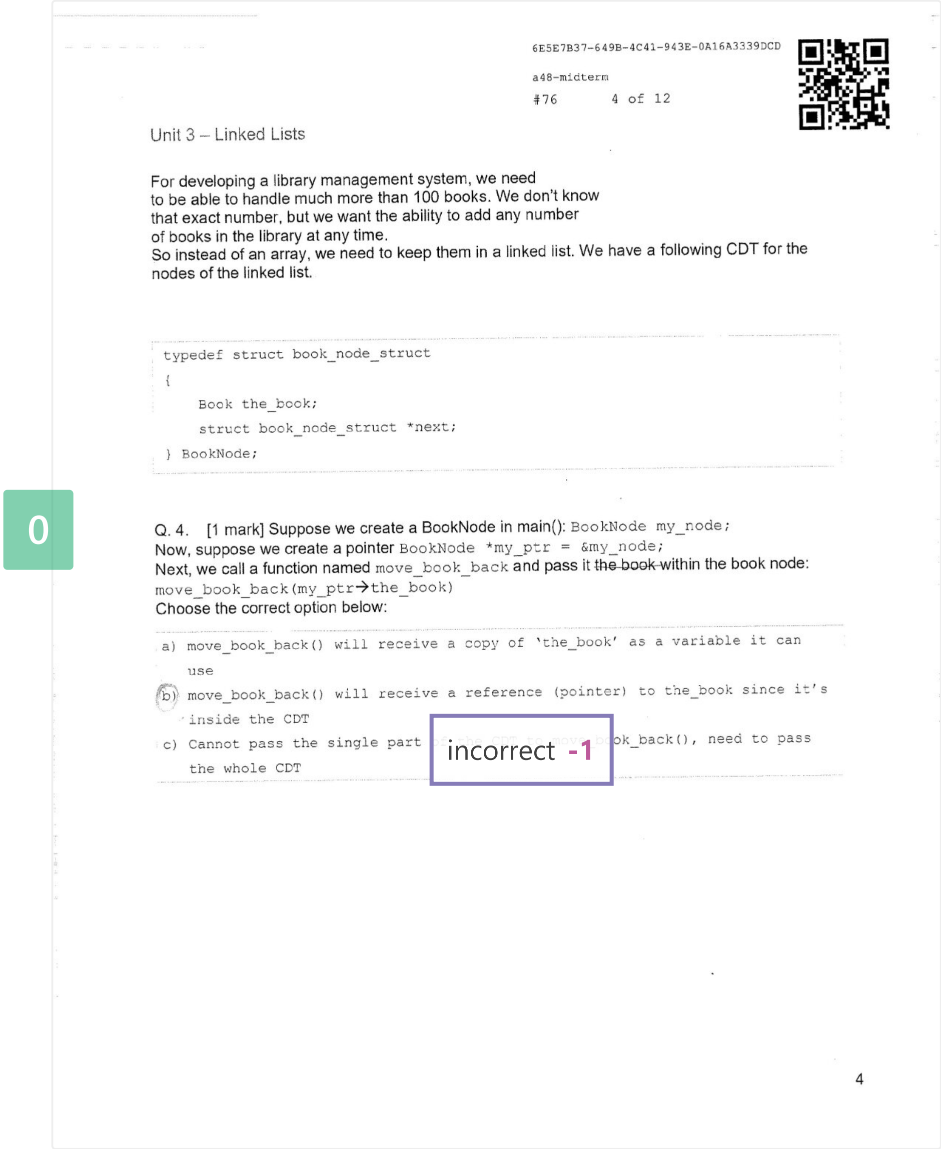
```
int main()
{
   Library the_lib;

   //write the code needed to initialize the_lib to contain 1 book and
   //set it up as described above.
   //This should take around 6 lines of code (not counting curly braces)
    the_lib . n-books = 5;
   strcpy(the_lib -> books[0]. book_name ,
    the_lib -> books[0]. type = 1;
    the_lib -> books[0]. copies = 5;
    the_lib -> books[0]. loc[0] = 5;
    the_lib -> books[0]. loc[1] = 20;
    the_lib -> books[0]. loc[2] = 4;
}
```

use calloc for books[0] -1

. not -> for the_lib -1

3

Q4 0



6E5E7B37-649B-4C41-943E-0A16A3339DCD

a48-midterm

#76 4 of 12

Unit 3 – Linked Lists

For developing a library management system, we need to be able to handle much more than 100 books. We don't know that exact number, but we want the ability to add any number of books in the library at any time.
So instead of an array, we need to keep them in a linked list. We have a following CDT for the nodes of the linked list.

```
typedef struct book_node_struct
{
    Book the_book;
    struct book_node_struct *next;
} BookNode;
```

Q. 4. [1 mark] Suppose we create a BookNode in main(): `BookNode my_node;`
Now, suppose we create a pointer `BookNode *my_ptr = &my_node;`
Next, we call a function named `move_book_back` and pass it ~~the book~~ within the book node:
`move_book_back(my_ptr→the_book)`
Choose the correct option below:

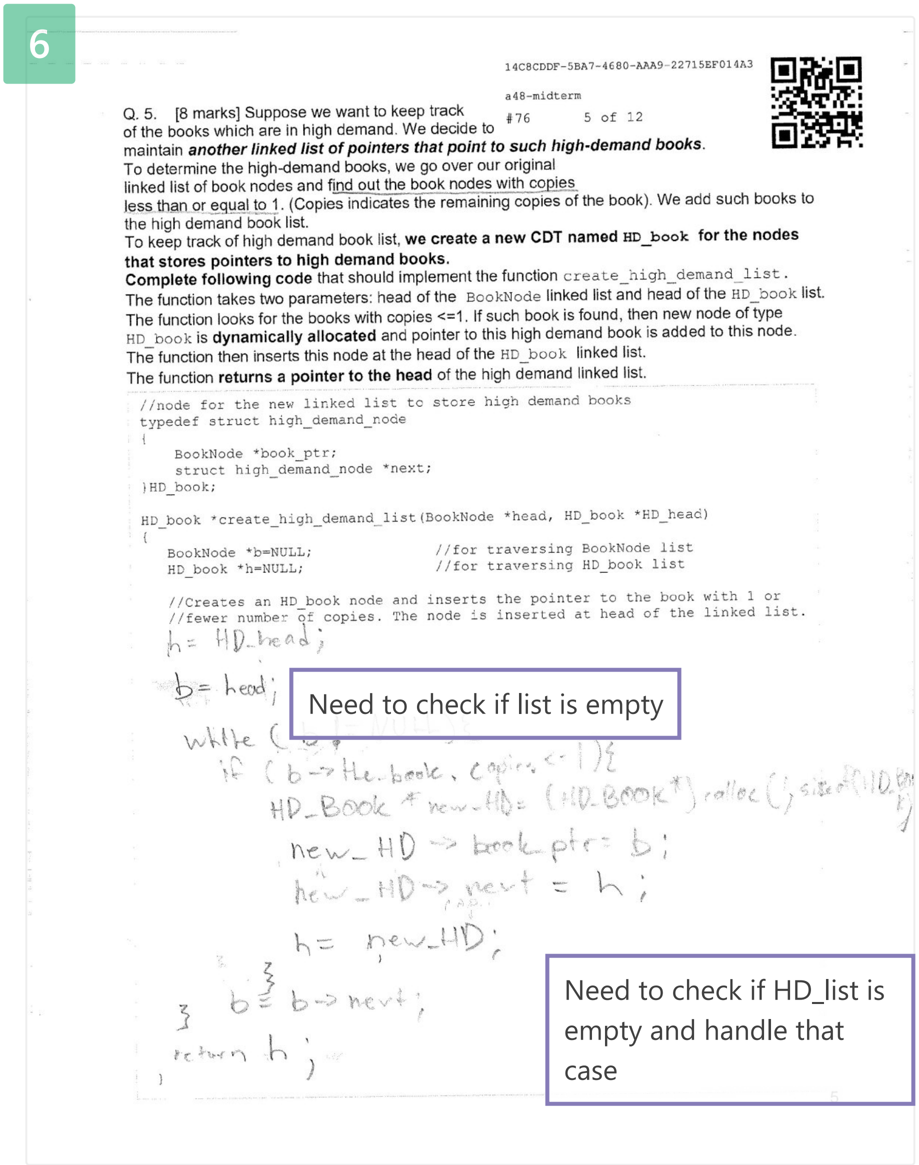
a) move_book_back() will receive a copy of 'the_book' as a variable it can use

[✓ b)] move_book_back() will receive a reference (pointer) to the_book since it's inside the CDT

c) Cannot pass the single part of the CDT to move_book_back(), need to pass the whole CDT

incorrect -1

Q5 6



14C8CDDF-5BA7-4680-AAA9-22715EF014A3

a48-midterm

#76 5 of 12

Q. 5. [8 marks] Suppose we want to keep track of the books which are in high demand. We decide to maintain ***another linked list of pointers that point to such high-demand books***. To determine the high-demand books, we go over our original linked list of book nodes and find out the book nodes with copies less than or equal to 1. (Copies indicates the remaining copies of the book). We add such books to the high demand book list.

To keep track of high demand book list, **we create a new CDT named `HD_book` for the nodes that stores pointers to high demand books.**

Complete following code that should implement the function `create_high_demand_list`.

The function takes two parameters: head of the `BookNode` linked list and head of the `HD_book` list.

The function looks for the books with copies <=1. If such book is found, then new node of type `HD_book` is **dynamically allocated** and pointer to this high demand book is added to this node.

The function then inserts this node at the head of the `HD_book` linked list.

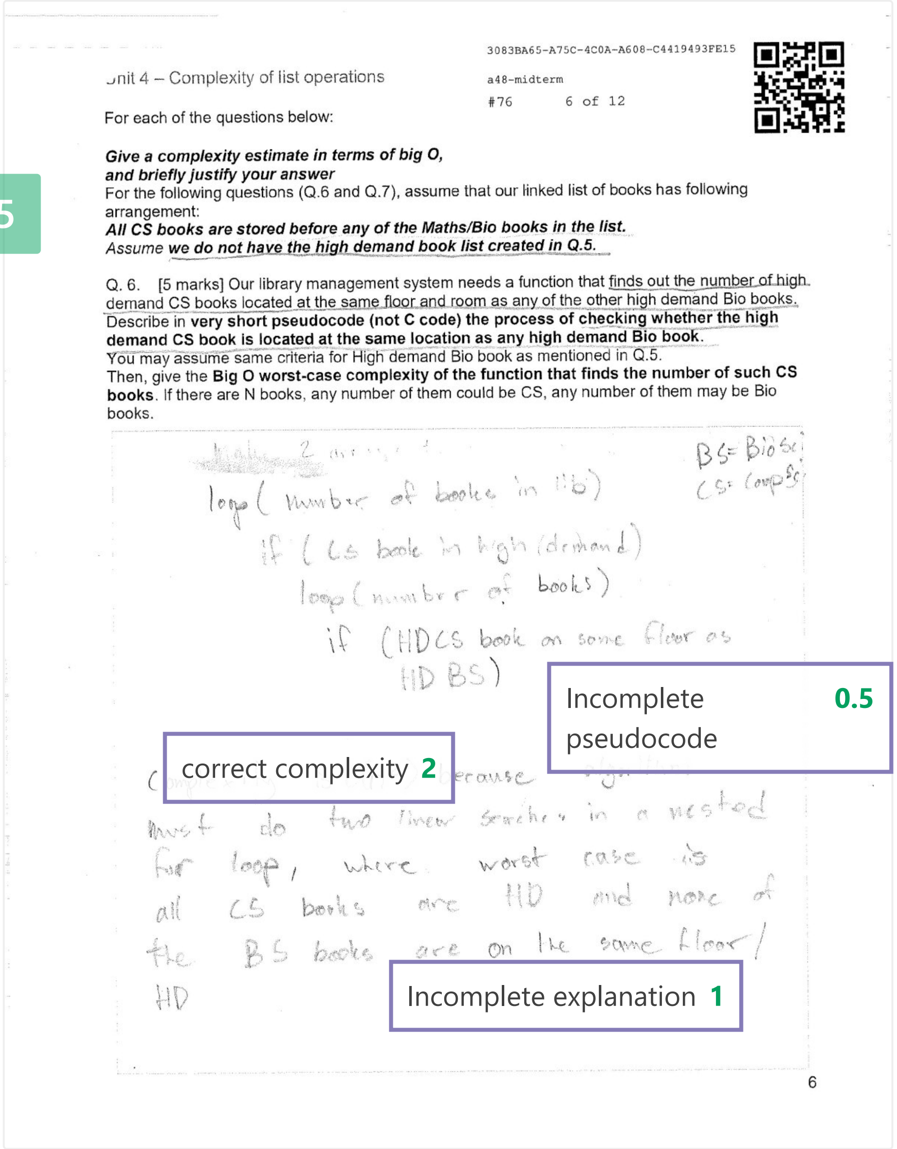
The function **returns a pointer to the head** of the high demand linked list.

```
//node for the new linked list to store high demand books
typedef struct high_demand_node
{
    BookNode *book_ptr;
    struct high_demand_node *next;
}HD_book;

HD_book *create_high_demand_list(BookNode *head, HD_book *HD_head)
{
   BookNode *b=NULL;                //for traversing BookNode list
   HD_book *h=NULL;                 //for traversing HD_book list

   //Creates an HD_book node and inserts the pointer to the book with 1 or
   //fewer number of copies. The node is inserted at head of the linked list.
   h= HD_head;

   b= head;

    while (b != NULL){
      if (b->He.book.copies <= 1){
        HD_Book *new_HD= (HD_BOOK*) calloc(1, sizeof(HD_Book));
         new_HD -> book_ptr= b;
         new_HD -> next = h;

         h= new_HD;
       }
     b = b->next;
   }
   return h;
}
```

Need to check if list is empty

Need to check if HD_list is empty and handle that case

Q6 3.5



3083BA65-A75C-4C0A-A608-C4419493FE15
a48-midterm
#76 6 of 12

Unit 4 – Complexity of list operations

For each of the questions below:

Give a complexity estimate in terms of big O, and briefly justify your answer

For the following questions (Q.6 and Q.7), assume that our linked list of books has following arrangement:

All CS books are stored before any of the Maths/Bio books in the list.

Assume ***we do not have the high demand book list created in Q.5.***

Q. 6. [5 marks] Our library management system needs a function that finds out the number of high demand CS books located at the same floor and room as any of the other high demand Bio books. Describe in **very short pseudocode (not C code) the process of checking whether the high demand CS book is located at the same location as any high demand Bio book**. You may assume same criteria for High demand Bio book as mentioned in Q.5. Then, give the **Big O worst-case complexity of the function that finds the number of such CS books**. If there are N books, any number of them could be CS, any number of them may be Bio books.

BS= Bio Sci
CS= Comp Sci

```
loop ( number of books in lib)
    if ( CS book in high (demand)
        loop ( number of books)
            if (HDCS book on same floor as
                HD BS)
```

Incomplete pseudocode 0.5

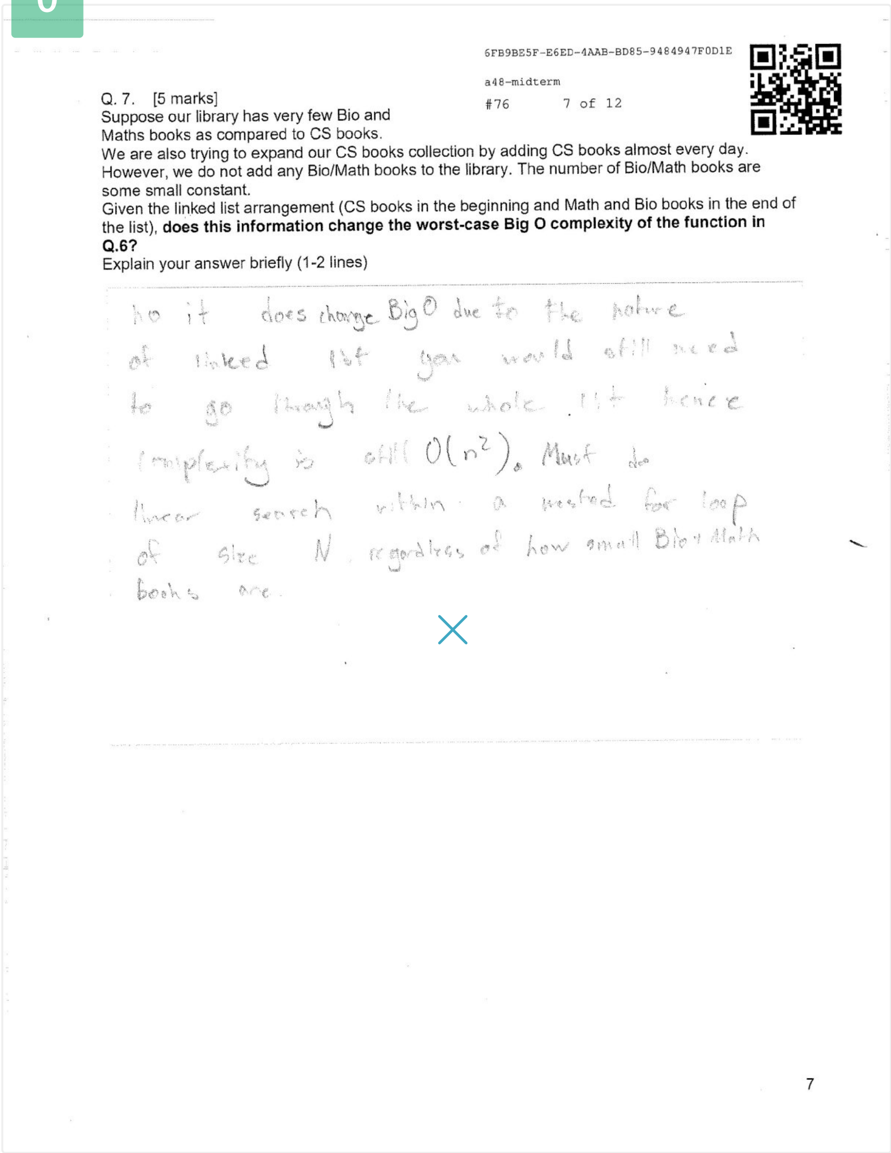
correct complexity 2

([illegible] because [illegible] must do two linear searches in a nested for loop, where worst case is all CS books are HD and none of the BS books are on the same floor / HD

Incomplete explanation 1

6

Q7 0



6FB9BE5F-E6ED-4AAB-BD85-9484947F0D1E

a48-midterm

#76 7 of 12

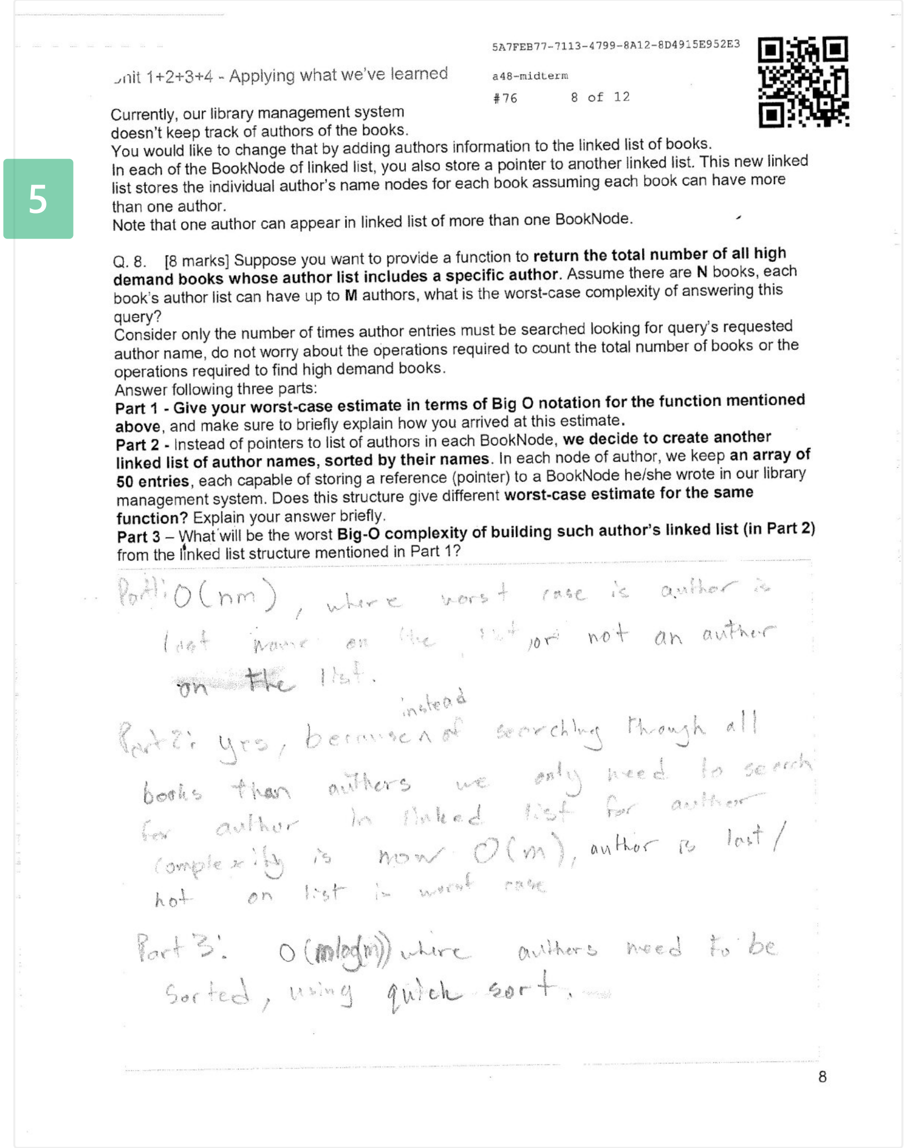
Q. 7. [5 marks]
Suppose our library has very few Bio and Maths books as compared to CS books.
We are also trying to expand our CS books collection by adding CS books almost every day.
However, we do not add any Bio/Math books to the library. The number of Bio/Math books are some small constant.
Given the linked list arrangement (CS books in the beginning and Math and Bio books in the end of the list), **does this information change the worst-case Big O complexity of the function in Q.6?**
Explain your answer briefly (1-2 lines)

no it does change Big O due to the nature of linked list you would still need to go through the whole list hence complexity is still $O(n^2)$. Must do linear search within a nested for loop of size N regardless of how small Bio + Math books are.

×

7

Q8 5



5A7FEB77-7113-4799-8A12-8D4915E952E3

a48-midterm

#76 8 of 12

Unit 1+2+3+4 - Applying what we've learned

Currently, our library management system doesn't keep track of authors of the books.
You would like to change that by adding authors information to the linked list of books.
In each of the BookNode of linked list, you also store a pointer to another linked list. This new linked list stores the individual author's name nodes for each book assuming each book can have more than one author.
Note that one author can appear in linked list of more than one BookNode.

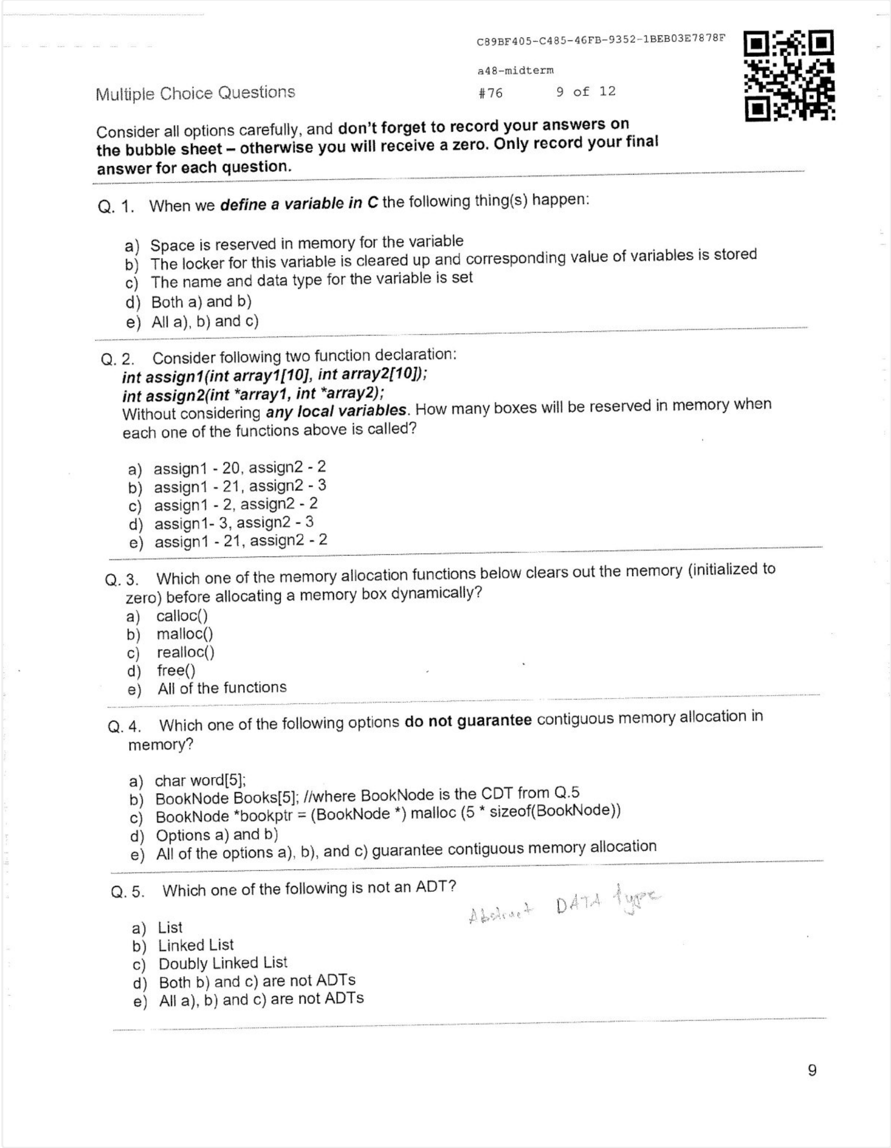
Q. 8. [8 marks] Suppose you want to provide a function to **return the total number of all high demand books whose author list includes a specific author**. Assume there are **N** books, each book's author list can have up to **M** authors, what is the worst-case complexity of answering this query?
Consider only the number of times author entries must be searched looking for query's requested author name, do not worry about the operations required to count the total number of books or the operations required to find high demand books.
Answer following three parts:
Part 1 - Give your worst-case estimate in terms of Big O notation for the function mentioned above, and make sure to briefly explain how you arrived at this estimate.
Part 2 - Instead of pointers to list of authors in each BookNode, **we decide to create another linked list of author names, sorted by their names**. In each node of author, we keep **an array of 50 entries**, each capable of storing a reference (pointer) to a BookNode he/she wrote in our library management system. Does this structure give different **worst-case estimate for the same function?** Explain your answer briefly.
Part 3 – What will be the worst **Big-O complexity of building such author's linked list (in Part 2)** from the linked list structure mentioned in Part 1?

Part1: O(nm), where worst case is author is last name on the list, or not an author on the list.

Part 2: yes, because instead of searching through all books than authors we only need to search for author in linked list for author complexity is now O(m), author is last / not on list is worst case

Part 3: O(mlog(m)) where authors need to be sorted, using quick sort.

8



Multiple Choice Questions

Consider all options carefully, and **don't forget to record your answers on the bubble sheet – otherwise you will receive a zero. Only record your final answer for each question.**

Q. 1. When we ***define a variable in C*** the following thing(s) happen:

a) Space is reserved in memory for the variable
b) The locker for this variable is cleared up and corresponding value of variables is stored
c) The name and data type for the variable is set
d) Both a) and b)
e) All a), b) and c)

Q. 2. Consider following two function declaration:
int assign1(int array1[10], int array2[10]);
int assign2(int *array1, int *array2);
Without considering ***any local variables***. How many boxes will be reserved in memory when each one of the functions above is called?

a) assign1 - 20, assign2 - 2
b) assign1 - 21, assign2 - 3
c) assign1 - 2, assign2 - 2
d) assign1- 3, assign2 - 3
e) assign1 - 21, assign2 - 2

Q. 3. Which one of the memory allocation functions below clears out the memory (initialized to zero) before allocating a memory box dynamically?
a) calloc()
b) malloc()
c) realloc()
d) free()
e) All of the functions

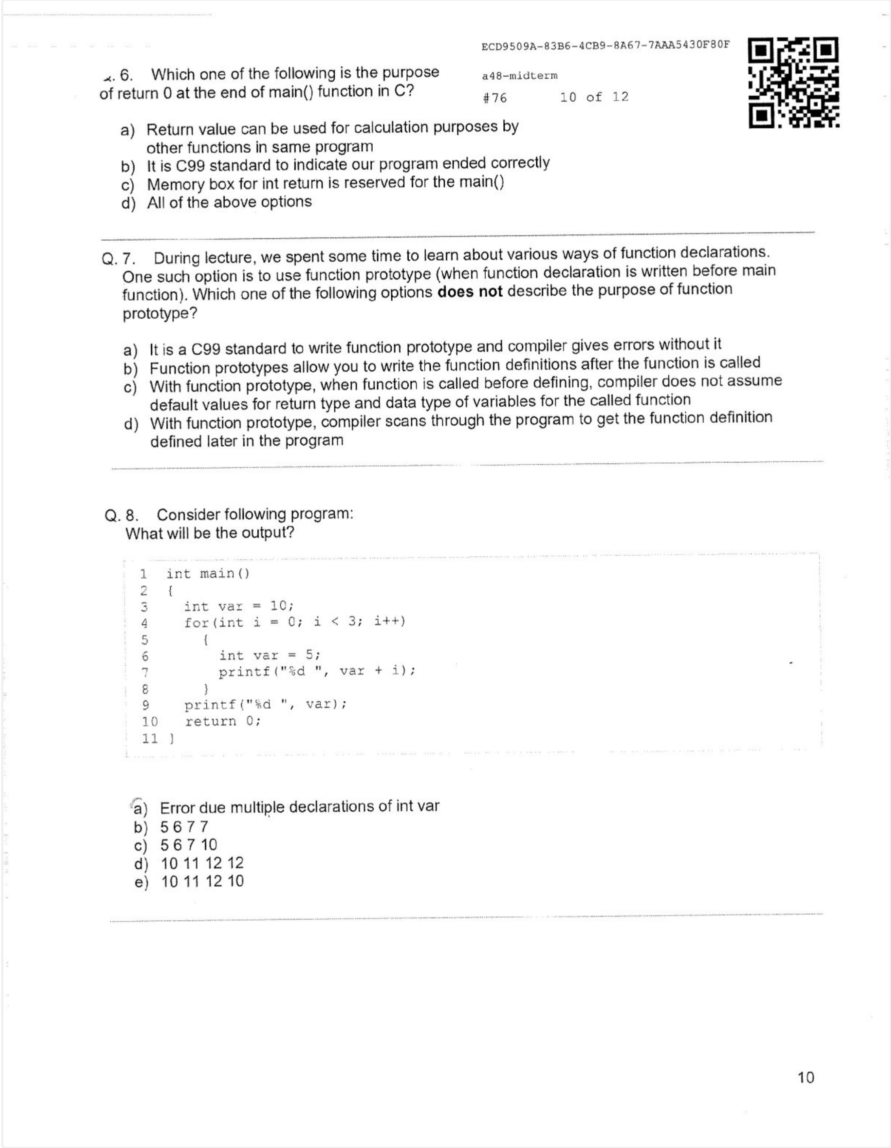
Q. 4. Which one of the following options **do not guarantee** contiguous memory allocation in memory?

a) char word[5];
b) BookNode Books[5]; //where BookNode is the CDT from Q.5
c) BookNode *bookptr = (BookNode *) malloc (5 * sizeof(BookNode))
d) Options a) and b)
e) All of the options a), b), and c) guarantee contiguous memory allocation

Q. 5. Which one of the following is not an ADT?

Abstract DATA type

a) List
b) Linked List
c) Doubly Linked List
d) Both b) and c) are not ADTs
e) All a), b) and c) are not ADTs



Q. 6. Which one of the following is the purpose of return 0 at the end of main() function in C?

a) Return value can be used for calculation purposes by other functions in same program
b) It is C99 standard to indicate our program ended correctly
c) Memory box for int return is reserved for the main()
d) All of the above options

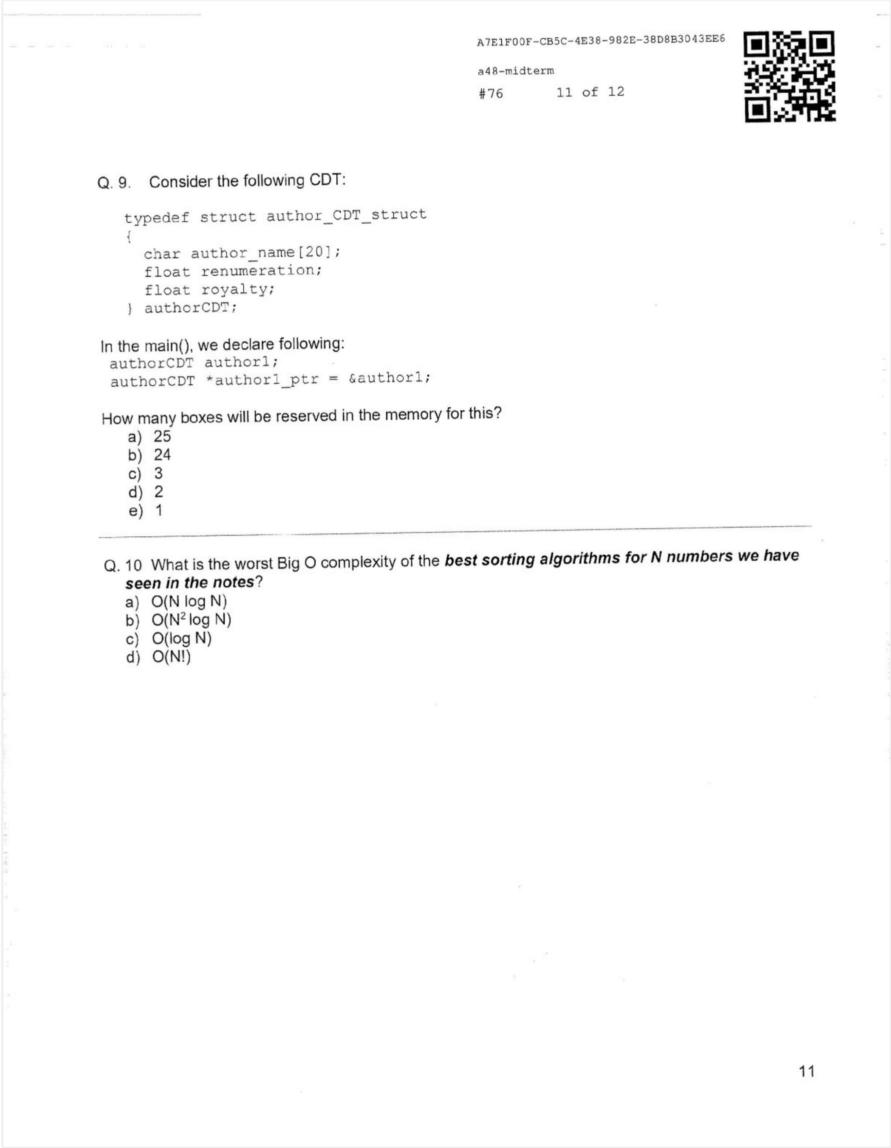
Q. 7. During lecture, we spent some time to learn about various ways of function declarations. One such option is to use function prototype (when function declaration is written before main function). Which one of the following options **does not** describe the purpose of function prototype?

a) It is a C99 standard to write function prototype and compiler gives errors without it
b) Function prototypes allow you to write the function definitions after the function is called
c) With function prototype, when function is called before defining, compiler does not assume default values for return type and data type of variables for the called function
d) With function prototype, compiler scans through the program to get the function definition defined later in the program

Q. 8. Consider following program:
What will be the output?

```
int main()
{
  int var = 10;
  for(int i = 0; i < 3; i++)
    {
      int var = 5;
      printf("%d ", var + i);
    }
  printf("%d ", var);
  return 0;
}
```

a) Error due multiple declarations of int var
b) 5 6 7 7
c) 5 6 7 10
d) 10 11 12 12
e) 10 11 12 10



A7E1F00F-CB5C-4E38-982E-38D8B3043EE6

a48-midterm

#76 11 of 12

Q. 9. Consider the following CDT:

```
typedef struct author_CDT_struct
{
  char author_name[20];
  float renumeration;
  float royalty;
} authorCDT;
```

In the main(), we declare following:

```
authorCDT author1;
authorCDT *author1_ptr = &author1;
```

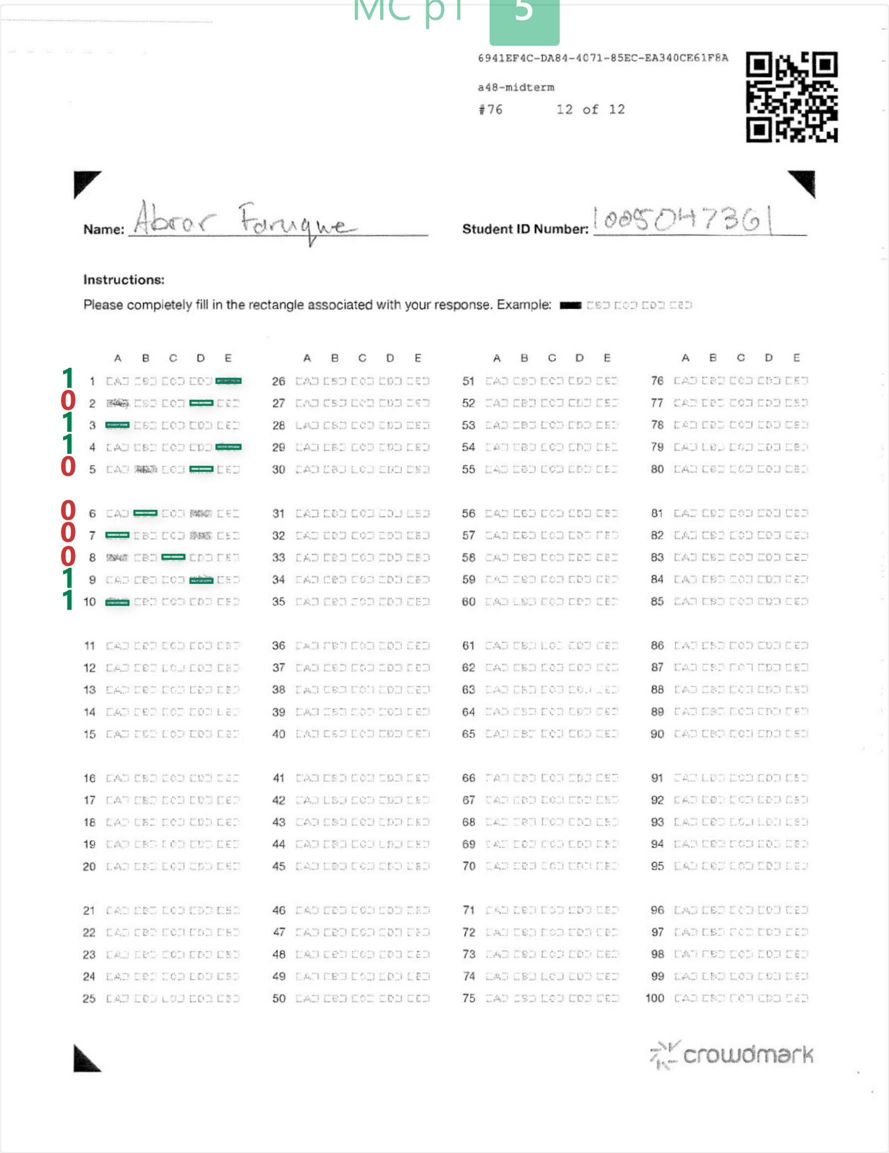
How many boxes will be reserved in the memory for this?

a) 25
b) 24
c) 3
d) 2
e) 1

Q. 10 What is the worst Big O complexity of the ***best sorting algorithms for N numbers we have seen in the notes***?

a) O(N log N)
b) O(N^2 log N)
c) O(log N)
d) O(N!)

11



MC p1 5

6941EF4C-DA84-4071-85EC-EA340CE61F8A

a48-midterm

#76 12 of 12

Name: Abror Faruque

Student ID Number: 1005047361

Instructions:

Please completely fill in the rectangle associated with your response. Example: ■ [B] [C] [D] [E]

		A	B	C	D	E		A	B	C	D	E		A	B	C	D	E		A	B	C	D	E
1	1	[A]	[B]	[C]	[D]	🟩	26	[A]	[B]	[C]	[D]	[E]	51	[A]	[B]	[C]	[D]	[E]	76	[A]	[B]	[C]	[D]	[E]
0	2	■	[B]	[C]	🟩	[E]	27	[A]	[B]	[C]	[D]	[E]	52	[A]	[B]	[C]	[D]	[E]	77	[A]	[B]	[C]	[D]	[E]
1	3	🟩	[B]	[C]	[D]	[E]	28	[A]	[B]	[C]	[D]	[E]	53	[A]	[B]	[C]	[D]	[E]	78	[A]	[B]	[C]	[D]	[E]
1	4	[A]	[B]	[C]	[D]	🟩	29	[A]	[B]	[C]	[D]	[E]	54	[A]	[B]	[C]	[D]	[E]	79	[A]	[B]	[C]	[D]	[E]
0	5	[A]	■	[C]	🟩	[E]	30	[A]	[B]	[C]	[D]	[E]	55	[A]	[B]	[C]	[D]	[E]	80	[A]	[B]	[C]	[D]	[E]
0	6	[A]	🟩	[C]	■	[E]	31	[A]	[B]	[C]	[D]	[E]	56	[A]	[B]	[C]	[D]	[E]	81	[A]	[B]	[C]	[D]	[E]
0	7	🟩	[B]	[C]	■	[E]	32	[A]	[B]	[C]	[D]	[E]	57	[A]	[B]	[C]	[D]	[E]	82	[A]	[B]	[C]	[D]	[E]
0	8	■	[B]	🟩	[D]	[E]	33	[A]	[B]	[C]	[D]	[E]	58	[A]	[B]	[C]	[D]	[E]	83	[A]	[B]	[C]	[D]	[E]
1	9	[A]	[B]	[C]	🟩	[E]	34	[A]	[B]	[C]	[D]	[E]	59	[A]	[B]	[C]	[D]	[E]	84	[A]	[B]	[C]	[D]	[E]
1	10	🟩	[B]	[C]	[D]	[E]	35	[A]	[B]	[C]	[D]	[E]	60	[A]	[B]	[C]	[D]	[E]	85	[A]	[B]	[C]	[D]	[E]
	11	[A]	[B]	[C]	[D]	[E]	36	[A]	[B]	[C]	[D]	[E]	61	[A]	[B]	[C]	[D]	[E]	86	[A]	[B]	[C]	[D]	[E]
	12	[A]	[B]	[C]	[D]	[E]	37	[A]	[B]	[C]	[D]	[E]	62	[A]	[B]	[C]	[D]	[E]	87	[A]	[B]	[C]	[D]	[E]
	13	[A]	[B]	[C]	[D]	[E]	38	[A]	[B]	[C]	[D]	[E]	63	[A]	[B]	[C]	[D]	[E]	88	[A]	[B]	[C]	[D]	[E]
	14	[A]	[B]	[C]	[D]	[E]	39	[A]	[B]	[C]	[D]	[E]	64	[A]	[B]	[C]	[D]	[E]	89	[A]	[B]	[C]	[D]	[E]
	15	[A]	[B]	[C]	[D]	[E]	40	[A]	[B]	[C]	[D]	[E]	65	[A]	[B]	[C]	[D]	[E]	90	[A]	[B]	[C]	[D]	[E]
	16	[A]	[B]	[C]	[D]	[E]	41	[A]	[B]	[C]	[D]	[E]	66	[A]	[B]	[C]	[D]	[E]	91	[A]	[B]	[C]	[D]	[E]
	17	[A]	[B]	[C]	[D]	[E]	42	[A]	[B]	[C]	[D]	[E]	67	[A]	[B]	[C]	[D]	[E]	92	[A]	[B]	[C]	[D]	[E]
	18	[A]	[B]	[C]	[D]	[E]	43	[A]	[B]	[C]	[D]	[E]	68	[A]	[B]	[C]	[D]	[E]	93	[A]	[B]	[C]	[D]	[E]
	19	[A]	[B]	[C]	[D]	[E]	44	[A]	[B]	[C]	[D]	[E]	69	[A]	[B]	[C]	[D]	[E]	94	[A]	[B]	[C]	[D]	[E]
	20	[A]	[B]	[C]	[D]	[E]	45	[A]	[B]	[C]	[D]	[E]	70	[A]	[B]	[C]	[D]	[E]	95	[A]	[B]	[C]	[D]	[E]
	21	[A]	[B]	[C]	[D]	[E]	46	[A]	[B]	[C]	[D]	[E]	71	[A]	[B]	[C]	[D]	[E]	96	[A]	[B]	[C]	[D]	[E]
	22	[A]	[B]	[C]	[D]	[E]	47	[A]	[B]	[C]	[D]	[E]	72	[A]	[B]	[C]	[D]	[E]	97	[A]	[B]	[C]	[D]	[E]
	23	[A]	[B]	[C]	[D]	[E]	48	[A]	[B]	[C]	[D]	[E]	73	[A]	[B]	[C]	[D]	[E]	98	[A]	[B]	[C]	[D]	[E]
	24	[A]	[B]	[C]	[D]	[E]	49	[A]	[B]	[C]	[D]	[E]	74	[A]	[B]	[C]	[D]	[E]	99	[A]	[B]	[C]	[D]	[E]
	25	[A]	[B]	[C]	[D]	[E]	50	[A]	[B]	[C]	[D]	[E]	75	[A]	[B]	[C]	[D]	[E]	100	[A]	[B]	[C]	[D]	[E]

crowdmark