HOMEWORK-5
Total Points: 60

1. [5 points] Consider the following program to insert a node "n" at a given position "pos" in a linked list "head", as discussed in class. Write this program without the "->" operator to access pointers to structure elements. Use the "." operator instead. Don't change the names of variables or the number of statements in this program.

```
struct node* insert_pos(struct node *head, struct node *n, int pos) {
  if (pos == 0) {
    n->next = head;
    return n;
  }

  struct node *tmp = head;
  int i = 0;

  while (i != pos - 1 && tmp) {
    tmp = tmp->next;
    i = i + 1;
  }

  if (tmp != NULL) {
    n->next = tmp->next;
    tmp->next = n;
  }

  return head;
}
```

2. [15 points] As discussed in class, in the geometric expansion solution for dynamic arrays, if after a deletion only N/4 elements are present in the array, where the array can store N elements, we create a new array of size N/2 elements, copy N/4 elements from the old array to the new array and delete the old array. Let's say initially, N elements are present in the array, and we delete all the elements one by one until the array is not empty; what would be the amortized cost of a delete operation? During the delete operation, we delete the last element by simply decrementing the number of elements by one.

3. [15 points] Consider the following incomplete recursive program to reverse a linked list. This program takes the head of a linked list as input and returns the head of the reversed list. If in the original list, the node at position 0 points to the node at position 1, the node at position 1 points to the node at position 2, and so on, in the reverse list, the node at position n-1 (in the original list) points to the node at position n-2 (in the original list), the node at position n-2 points to the node at position n-3, and so on. The head of the reversed linked list is the node at position n-1 in the original list. Write the code for

the MISSING-BLOCK such that this program correctly reverses the linked list and returns the head of the reversed list.

```
struct node {
    int val;
    struct node *next;
};

struct node *reverse_list(struct node *head) {
  if (head == NULL || head->next == NULL) {
    return head;
  }
  struct node *reversed_list_head = reverse_list(head->next);

  // MISSING-BLOCK
  // WRITE-YOUR-CODE-HERE

  return reversed_list_head;
}
```

4. [25 points] Write an implementation of the delete_min_max routine below that deletes the maximum and minimum element from a linked list starting at "head" and returns the new head (if applicable) along with the deleted nodes corresponding to the minimum and maximum values. You can assume that all the linked list elements have distinct values. If the linked list contains only one element, the same element is considered both minimum and maximum. You are not allowed to call any function in your implementation. You can iterate through all the elements in the linked list exactly once in your implementation. Make sure that there are no syntax errors in your implementation.

```
struct node {
  int val;
  struct node *next;
};

struct delete_info {
  struct node *head; // head of the new linked list
  struct node *min;   // the deleted node that contains minimum value
  struct node *max;  // the deleted node that contains maximum value
};

/* delete nodes with minimum and maximum values and return the updated
head (if applicable) along with and deleted nodes using a return value
of type struct delete_info
*/
struct delete_info delete_min_max(struct node *head);
```

In addition to the implementation, also run your program for different inputs as given below and write the contents of the list after delete_min_max returns:

- What is the output list when the input list is 1 5 3 2 4
- What is the output list when the input list is 5 1 3 2 4
- What is the output list when the input list is 4 1 5 2 3
- What is the output list when the input list is 4 5 1 2 3
- What is the output list when the input list is 4 5 3 2 1
- What is the output list when the input list is 5 4 3 2 1