Exam Revision

```c
// Binary Search Tree search function

struct node *binaryTreeSearch(struct Node *root, int key){
    if (root == NULL) {
        return NULL;
    }
    if (root->key == key) {
        return root;
    }
    if (root->key > key) {
        return binaryTreeSearch(root->left, key);
    }
    else {
        return binaryTreeSearch(root->right, key);
    }
};
```

```c
// Linked List Recursive Search function

struct node {
    int key;
    struct node *next;
};

struct node *searchList(struct node *head, int key){
    if (head == NULL) {
        return NULL;
    }
    if (head->key == key) {
        return head;
    }
    return linkedListSearch(head->next, key);
};
```

```c
//Main function

int main(void) {
    struct node *head = NULL;
    struct node *node = NULL;
    int key = 0;
    int i = 0;
    int *keys[5] = {1, 3, 7, 12 15};

    for (i = 0; i < 5; i++) {
        node = (struct node *)malloc(sizeof(struct node));
        if (node == NULL) {
            printf("Error\n");
            return 1;
        }
        node->key = keys[i];
        node->next = head;
        head = node;
    }

    scanf("%d", &key);

    node = searchList(head, key);
    if (node == NULL) {
        printf("%d is not found\n", key);
    }
    else {
        printf("%d is found\n", key);
    }
}
```

Tree traversal question 1

```
                49
            /        \
         28           83
       /    \       /    \
     18     40     71     97
    /  \   /  \   /  \   /  \
  11   19 32  44 69  72 92  99
```

In order traversal: 11 18 19 28 32 40 44 49 69 71 72 83 92 97 99

Pre order traversal: 49 28 18 11 19 40 32 44 83 71 69 72 97 92 99

Post order traversal: 11 19 18 32 44 40 28 69 72 71 92 99 97 83 49

Tree traversal question 2

```
        6
      /   \
     3    34
         /
        17
       /  \
      16  19
     /      \
    10      23
```

Post order traversal: 3 10 16 23 19 17 34 6

Tree traversal question 3

```
        15
      /    \
     6     34
   /  \    /
  2   10  17
   \     /  \
    3   16  19
```

In order traversal: 2 3 6 10 15 16 17 19 34

Tree traversal question 4

```
            A
          /   \
         B     C
        / \   / \
       D   E F   G
      / \
     H   J
```

In order traversal: A B C D E F G H J

Pre order traversal: A B D H J E C F G

Post order traversal: H J D E B F G C A

```
    // Insert 25 random integers from 0 to 100 in order in a linked list
    // Then the program should calculate the sum of the elements and the
floating point average of the elements

    #include <stdio.h>
    #include <stdlib.h>
    #include <math.h>
    #include <time.h>

    struct node {
        int key;
        struct node *next;
    };

    int main(void) {
        struct node *head = NULL;
        struct node *node = NULL;
        int sum = 0;
        int average = 0;
        int keys[25] = {0};

        srand(time(NULL));

        for (int i = 0; i < 25; i++) {
            keys[i] = rand() % 100;
            printf("%d ", keys[i]);
        }
        printf("\n");

        for (int i = 0; i < 25; i++) {
            node = (struct node *)malloc(sizeof(struct node));
            if (node == NULL) {
                printf("Error\n");
                return 1;
            }
            node->key = keys[i];
            node->next = head;
            head = node;
        }

        node = head;
        while (node != NULL) {
            sum += node->key;
            node = node->next;
        }

        average = sum / 25;

        printf("Sum: %d\n", sum);
        printf("Average: %d\n", average);
    }
```

```
    //Write a program that creates a linked list if 10 characters, then
create a copy of the list in reverse order.

    #include <stdio.h>
    #include <stdlib.h>

    struct node {
        char key;
        struct node *next;
    };

    int main(void) {
        struct node *head = NULL;
        struct node *node = NULL;
        struct node *reverse_head = NULL;
        struct node *reverse_node = NULL;
        char *keys[10] = {"a", "b", "c", "d", "e", "f", "g", "h", "i",
"j"};
```

```
        for (int i = 0; i < 10; i++) {
            node = (struct node *)malloc(sizeof(struct node));
            if (node == NULL) {
                printf("Error\n");
                return 1;
            }
            node->key = *keys[i];
            node->next = head;
            head = node;
        }

        node = head;
        while (node != NULL) {
            reverse_node = (struct node *)malloc(sizeof(struct node));
            if (reverse_node == NULL) {
                printf("Error\n");
                return 1;
            }
            reverse_node->key = node->key;
            reverse_node->next = reverse_head;
            reverse_head = reverse_node;
            node = node->next;
        }

        printf("Before reverse: ");

        node = head;
        while (node != NULL) {
            printf("%c", node->key);
            node = node->next;
        }
        printf("\n");

        printf("After reverse: ");

        node = reverse_head;
        while (node != NULL) {
            printf("%c", node->key);
            node = node->next;
        }
        printf("\n");
    }
```