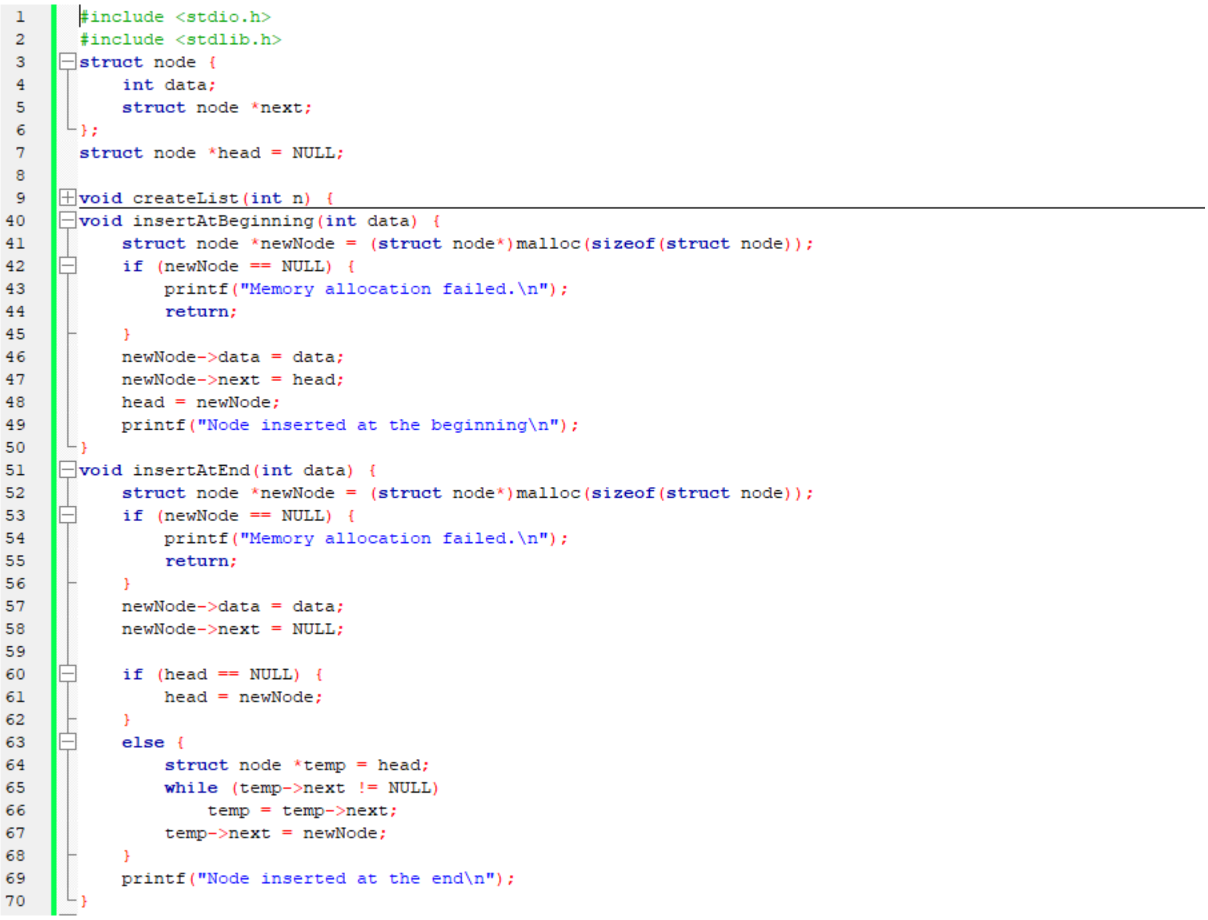
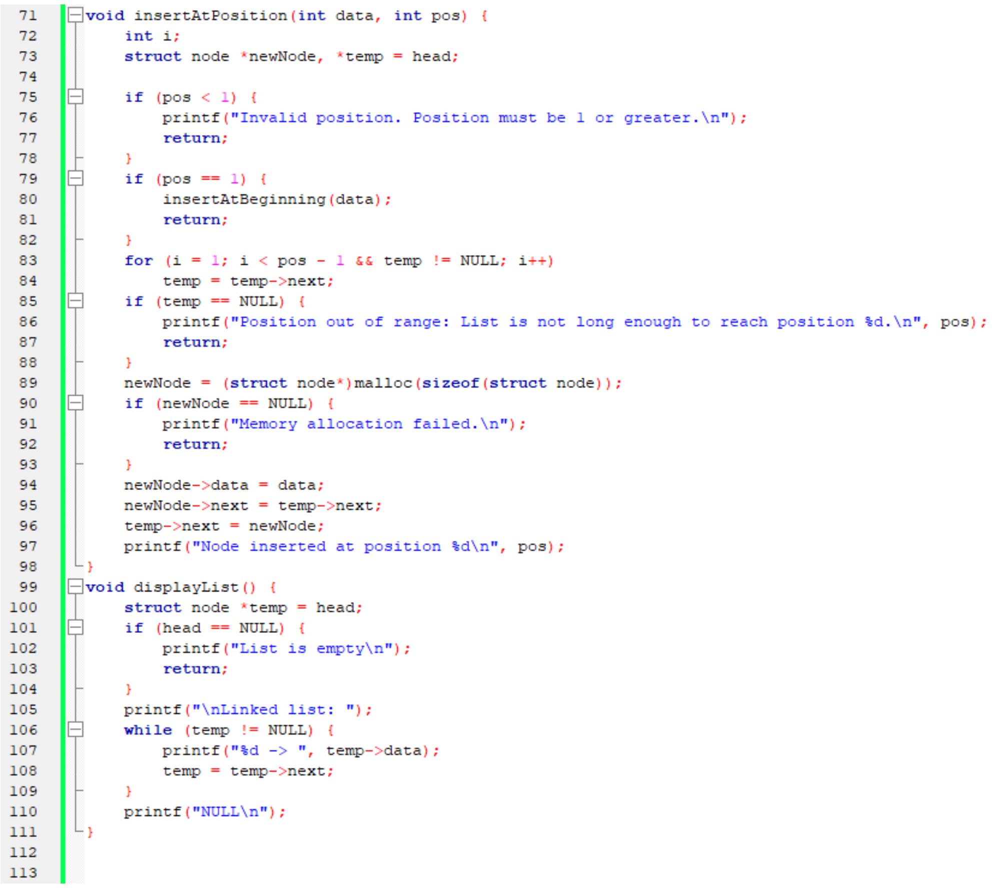
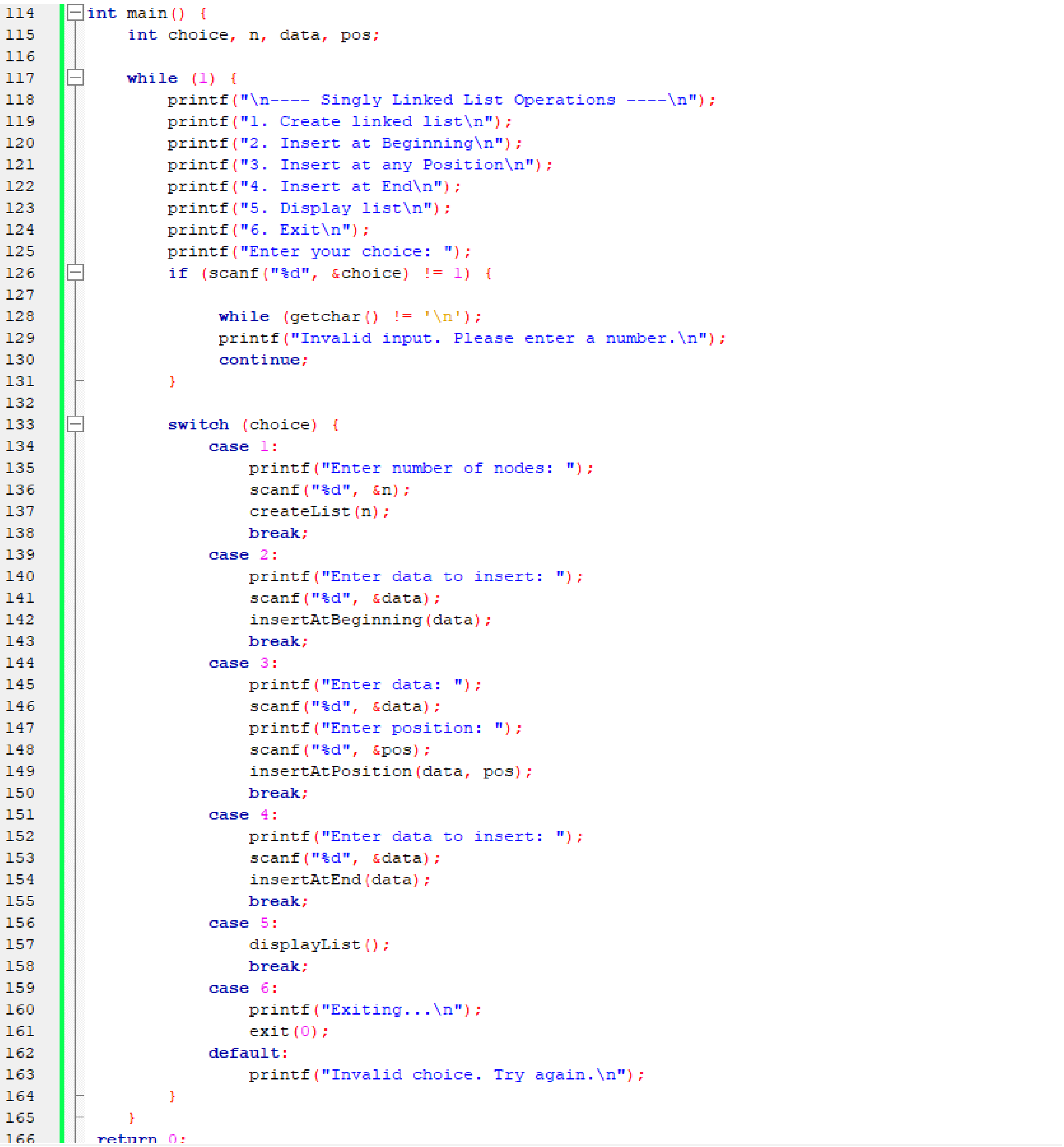


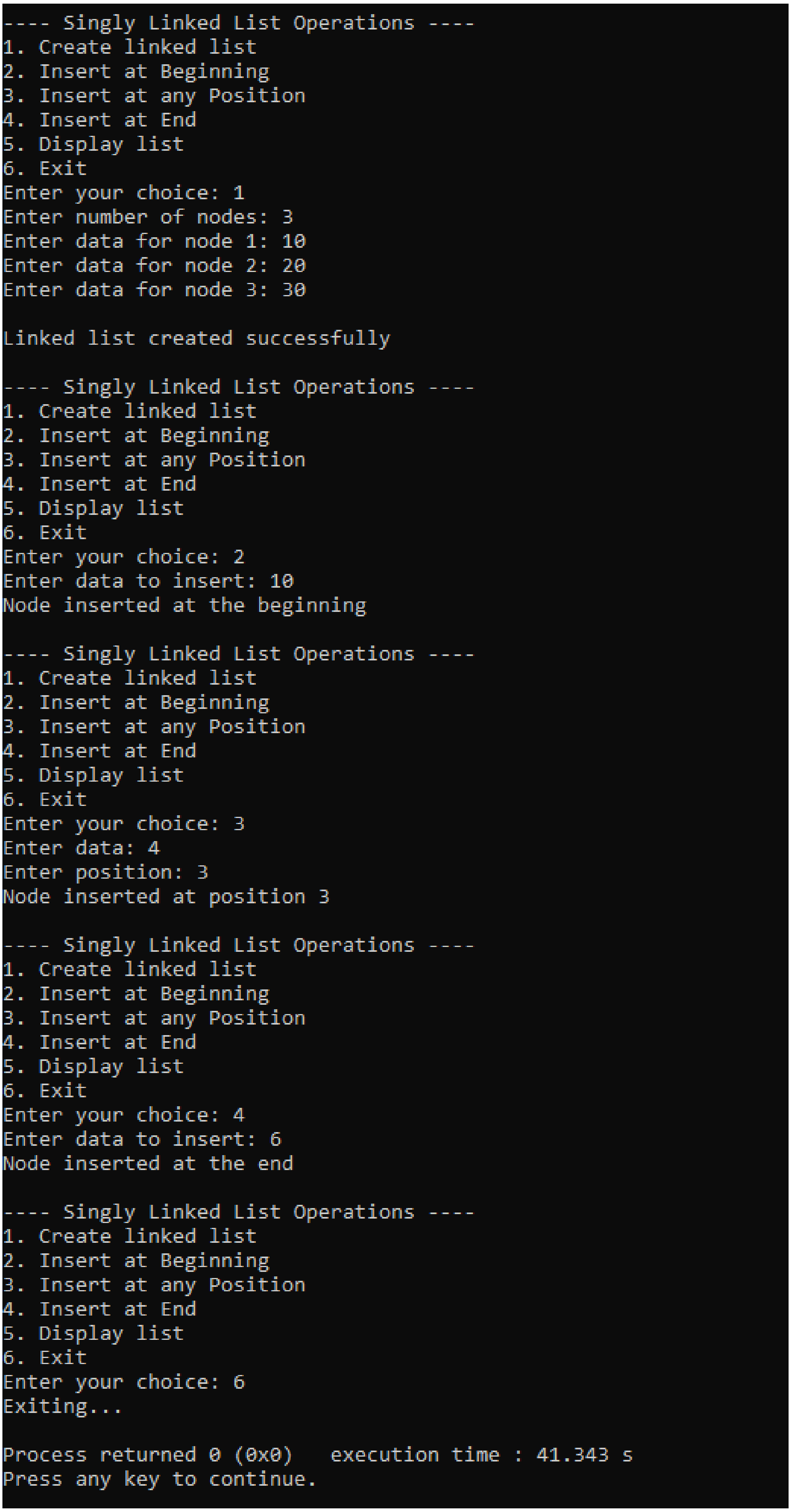
```c
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node *next;
};
struct node *head = NULL;

void createList(int n) {
void insertAtBeginning(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = head;
    head = newNode;
    printf("Node inserted at the beginning\n");
}
void insertAtEnd(int data) {
    struct node *newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = NULL;

    if (head == NULL) {
        head = newNode;
    }
    else {
        struct node *temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
    printf("Node inserted at the end\n");
}
```



```c
void insertAtPosition(int data, int pos) {
    int i;
    struct node *newNode, *temp = head;

    if (pos < 1) {
        printf("Invalid position. Position must be 1 or greater.\n");
        return;
    }
    if (pos == 1) {
        insertAtBeginning(data);
        return;
    }
    for (i = 1; i < pos - 1 && temp != NULL; i++)
        temp = temp->next;
    if (temp == NULL) {
        printf("Position out of range: List is not long enough to reach position %d.\n", pos);
        return;
    }
    newNode = (struct node*)malloc(sizeof(struct node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        return;
    }
    newNode->data = data;
    newNode->next = temp->next;
    temp->next = newNode;
    printf("Node inserted at position %d\n", pos);
}
void displayList() {
    struct node *temp = head;
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    printf("\nLinked list: ");
    while (temp != NULL) {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}


```



```c
int main() {
    int choice, n, data, pos;

    while (1) {
        printf("\n---- Singly Linked List Operations ----\n");
        printf("1. Create linked list\n");
        printf("2. Insert at Beginning\n");
        printf("3. Insert at any Position\n");
        printf("4. Insert at End\n");
        printf("5. Display list\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        if (scanf("%d", &choice) != 1) {

             while (getchar() != '\n');
             printf("Invalid input. Please enter a number.\n");
             continue;
        }

        switch (choice) {
            case 1:
                printf("Enter number of nodes: ");
                scanf("%d", &n);
                createList(n);
                break;
            case 2:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 3:
                printf("Enter data: ");
                scanf("%d", &data);
                printf("Enter position: ");
                scanf("%d", &pos);
                insertAtPosition(data, pos);
                break;
            case 4:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                displayList();
                break;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Try again.\n");
        }
    }
 return 0;
```



```
---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 1
Enter number of nodes: 3
Enter data for node 1: 10
Enter data for node 2: 20
Enter data for node 3: 30

Linked list created successfully

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 2
Enter data to insert: 10
Node inserted at the beginning

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 3
Enter data: 4
Enter position: 3
Node inserted at position 3

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 4
Enter data to insert: 6
Node inserted at the end

---- Singly Linked List Operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any Position
4. Insert at End
5. Display list
6. Exit
Enter your choice: 6
Exiting...

Process returned 0 (0x0)   execution time : 41.343 s
Press any key to continue.
```

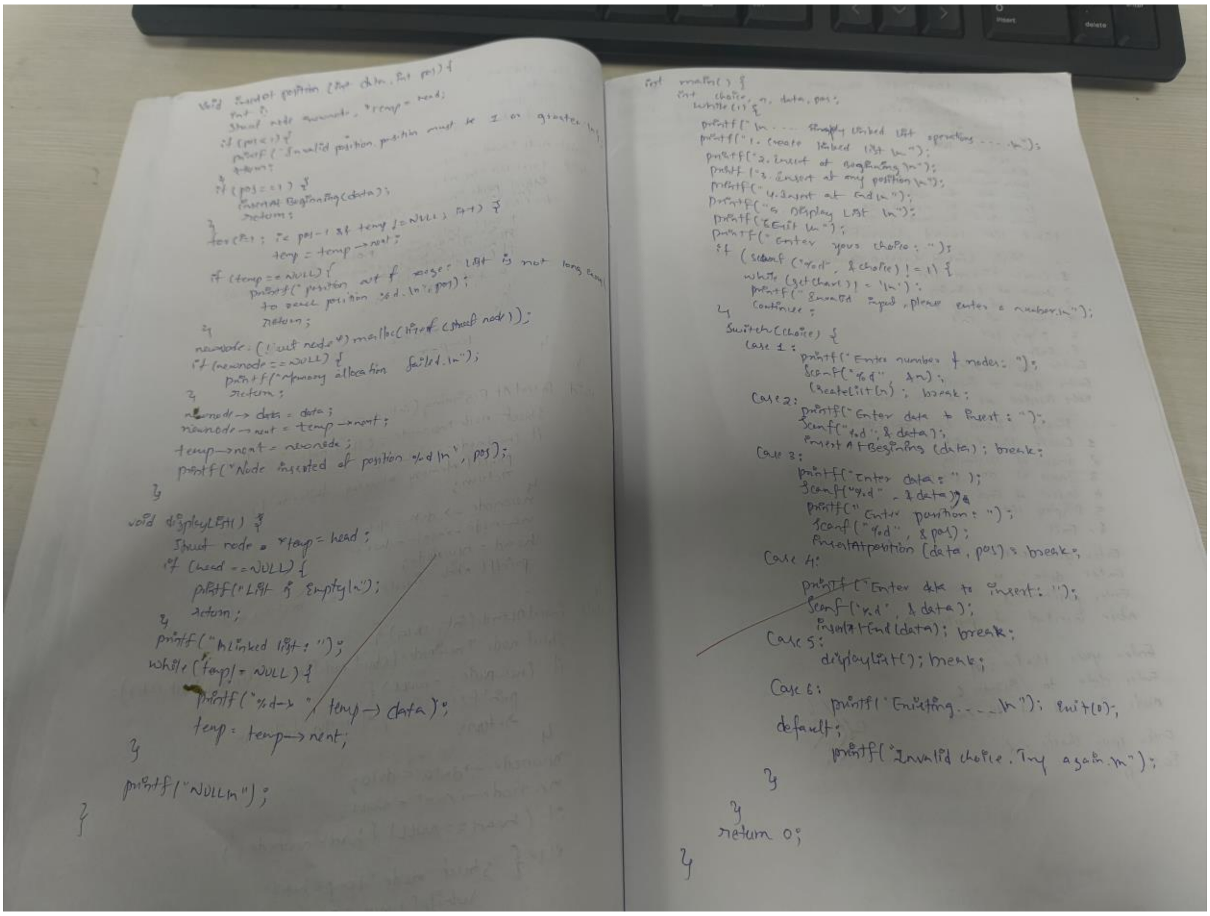
Lab program-4

(a) WAP to implement single linked list with the following operations

- → Create a Linked list
- → Insertion of a node at
  - → first position
  - → any position
  - → End position
- → Display the contents of linked lists

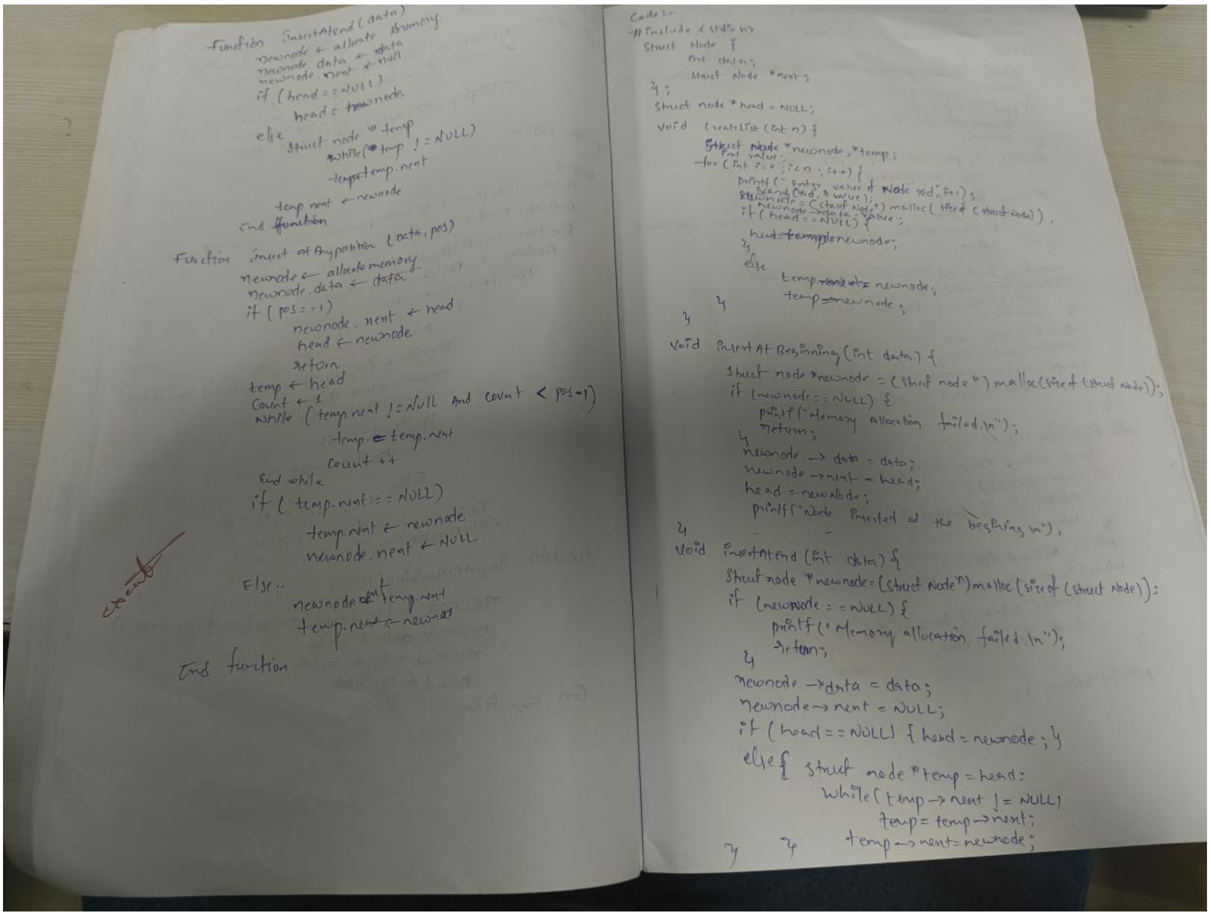
pseudo code :-

```
BEGIN
  Define Structure Node with contents data, next pointer
  Declare head pointer to Null
  Function CreateLinkedList(n)
    Declare i, ~~Data~~, temp, newNode
      For i←1 To n DO
               print "Enter value:"
                  Read value from input
               newnode ← (structNode*)malloc( sizeof(struct node))
               newnode.data ← data
               newnod.next ← null
               if head == NULL
                    head ← temp ← newnode
               Else
                    temp.next ← newnode
         END if             temp ← newnode
       END Function.

Function InsertAtBegining(Data)
         newnode ← (Struct Node*)malloc(sizeof(struct Node))
         newnode.data ← data
         newnod.next ← head.
         head ← newnode
End Function.
```



```
void insertAtposition (int data, int pos) {
    int i;
    struct node *newnode, *temp = head;
    if (pos<1) {
      printf("Invalid position. position must be 1 or greater.\n");
      return;
    }
    if (pos==1) {
      insertAtBeginning(data);
      return;
    }
    for(i=1; i< pos-1 && temp != NULL; i++) {
          temp = temp->next;
    if (temp == NULL) {
       printf("position out of range. List is not long enough
         to reach position %d.\n", pos);
         return;
    }
  newnode = (struct node*) malloc(sizeof (struct node));
  if (newnode == NULL) {
     printf("Memory allocation failed.\n");
  }  return;
  newnode -> data = data;
  newnode -> nent = temp -> nent;
  temp->nent = newnode;
  printf("Node inserted at position %d \n", pos);
 }
void displayList() {
   struct node * temp = head;
   if (head == NULL) {
     printf("List is Empty\n");
     return;
   }
   printf("\nLinked list : ");
   while (temp != NULL) {
       printf("%d-> ", temp-> data);
       temp = temp-> nent;
  }
 printf("NULL\n");
}
```

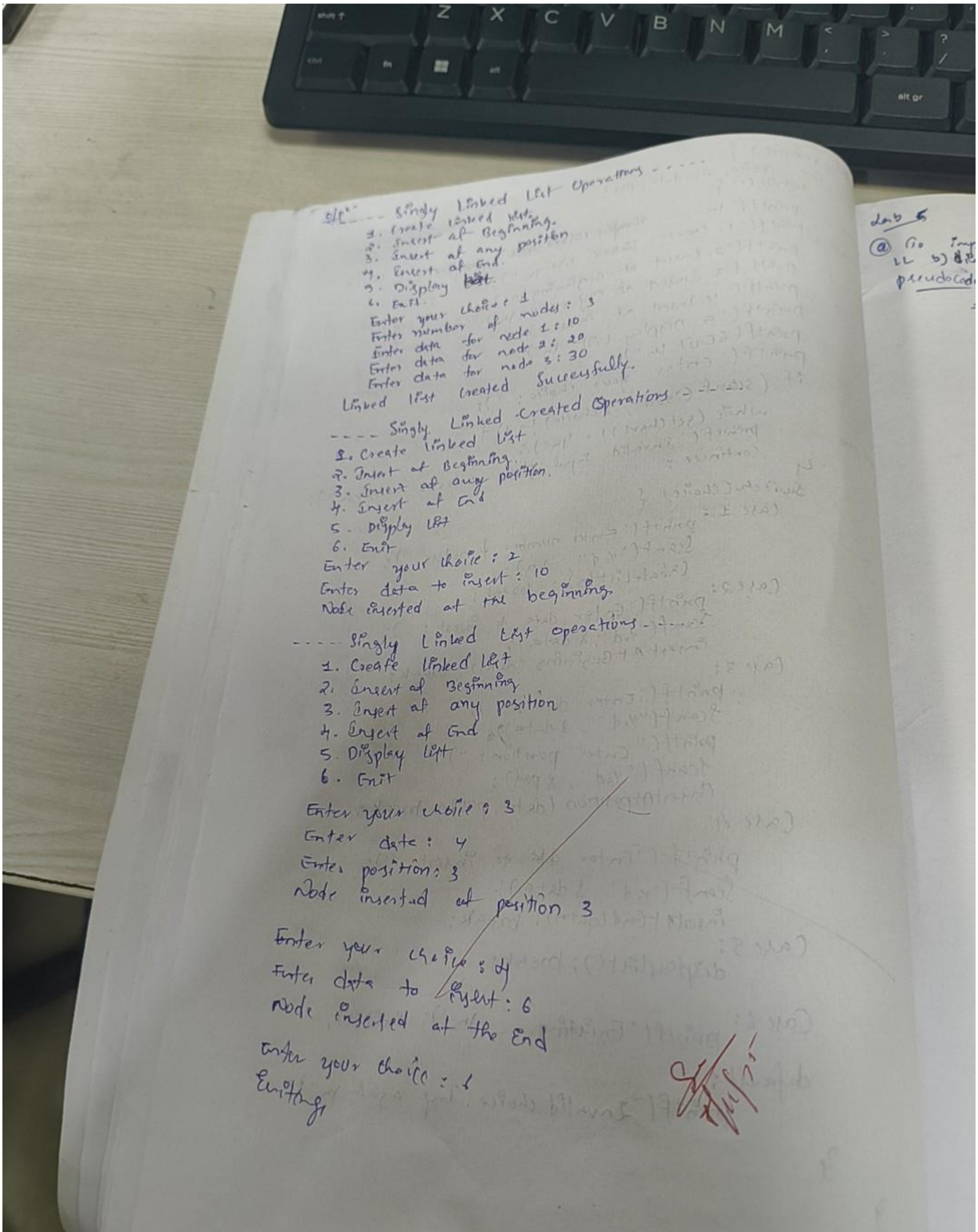
```
int main() {
 int choice, n, data, pos;
   while (1) {
    printf("\n.... Singly Linked List operations ....\n");
    printf("1. Create linked list \n");
    printf("2. Insert at Beginning \n");
    printf("3. Insert at any position \n");
    printf("4. Insert at End \n");
    printf("5 Display List \n");
    printf("6 Exit \n");
    printf("Enter your choice : ");
    if (scanf ("%d", &choice) != 1) {
      while (getchar() != '\n');
       printf("Invalid input, please enter a number.\n");
    }  Continue;
    Switch(choice) {
      Case 1 :
           printf("Enter number of nodes: ");
           scanf("%d", &n);
           CreateList(n); break;
      Case 2:
           printf("Enter data to insert : ");
           scanf("%d", & data);
           insertAtBeginning (data); break;
      Case 3:
           printf("Enter data: ");
           scanf("%d", & data);
           printf(" Enter position : ");
           scanf ("%d", & pos);
           insertAtposition (data, pos); break;
      Case 4:
           printf("Enter data to insert: ");
           scanf("%d", & data);
           insertAtEnd(data); break;
      Case 5:
           displayList(); break;
      Case 6:
           printf("Exiting....\n"); exit(0);
      default:
           printf("Invalid choice. Try again.\n");
      }
    }
   return 0;
 }
```



```
Function InsertAtEnd (data)
     newnode ← allocate memory
     newnode. data ← data
     newnode. next ← null
     if (head == NULL)
          head = newnode
     else
        struct node * temp
             while( temp != NULL)
             temp = temp. next
       temp next ← newnode
   end function

Function Insert at Anyposition (data, pos)
   newnode ← allocate memory
   newnode. data ← data
   if (pos == 1)
        newnode. next ← head
        head ← newnode
        return
   temp ← head
   Count ← 1
   while (temp next != Null And count < pos-1)
            temp. = temp. next
            count ++
   End while
   if ( temp. next == NULL)
        temp. next ← newnode
        newnode. next ← NULL
   Else :-
        newnode.next ← temp. next
        temp. next ← newnode
End function
```

Code :-

```
#include <stdio.h>
  Struct Node {
        int data;
        struct Node *next;
  };
  Struct node * head = NULL;
  void  CreateList (int n) {
        Struct Node *newnode, *temp;
          int value;
       for (int i=0 ; i<n ; i++) {
            printf(" Enter value of Node %d", i+1);
              Scanf ("%d", & value);
            newnode = (struct Node*) malloc( sizeof (struct node));
              newnode->data = value;
            if ( head == NULL) {
             head = newnode;
            }
            else
                  temp.next = newnode;
                  temp = newnode;
       }
   }
  void  insert At Beginning (int data) {
        Struct node *newnode = (Struct node *) malloc(sizeof (Struct node));
        if (newnode == NULL) {
           printf("Memory Allocation failed.\n");
           return;
        }
        newnode -> data = data;
        newnode -> next = head;
        head = newNode;
         printf("Node inserted at the beginning \n"),
  }
  void  insertAtEnd (int data) {
       Struct node *newnode= (struct Node*) malloc (sizeof (struct Node));
       if (newnode == NULL) {
           printf(" Memory allocation failed.\n");
           return;
       }
       newnode -> data = data;
       newnode -> next = NULL;
       if ( head == NULL) { head = newnode ; }
       else { struct node *temp = head;
                 While( temp -> next != NULL)
                      temp = temp -> next;
               temp -> next=newnode;
       }    }
```



```
o/p:- --- Singly Linked List Operations -----
 1. Create Linked list.
 2. Insert at Beginning.
 3. Insert at any position
 4. Insert at End.
 5. Display ~~list~~.
 6. Exit.
 Enter your choice: 1
 Enter number of nodes : 3
 Enter data for node 1 : 10
 Enter data for node 2 : 20
 Enter data for node 3 : 30
Linked list created Successfully.

---- Singly Linked Created Operations ----
1. Create linked list
2. Insert at Beginning.
3. Insert at any position.
4. Insert at End
5. Display list
6. Exit
Enter your choice : 2
Enter data to insert : 10
Node inserted at the beginning.

---- Singly Linked List operations ----
1. Create linked list
2. Insert at Beginning
3. Insert at any position
4. Insert at End
5. Display list
6. Exit

Enter your choice : 3
Enter date : 4
Enter position : 3
Node inserted at position 3

Enter your choice : 4
Enter data to insert : 6
Node inserted at the End

Enter your choice : 6
Exiting
```

Lab 5

@ Go Imp
LL 3)
pseudocod