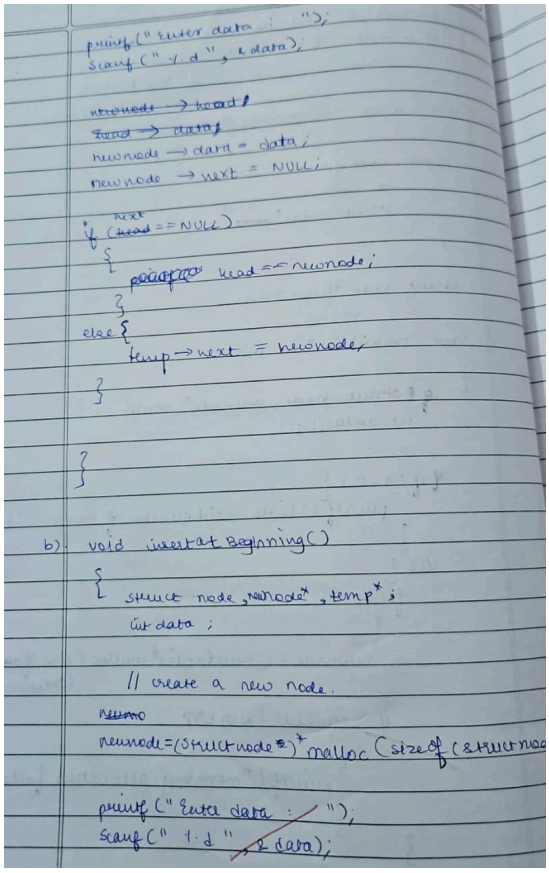
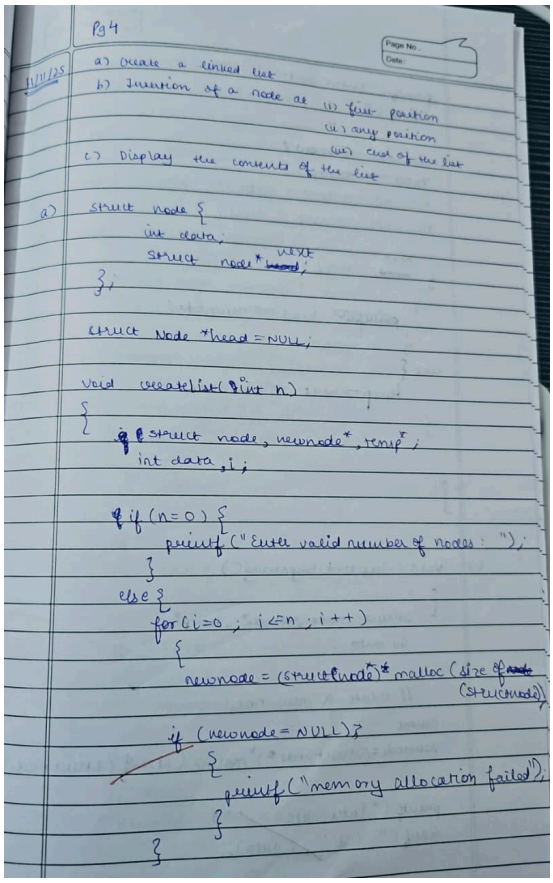
OUTPUT

```
 LINKED LIST OPERATIONS :
1. Create List
2. Insert at Beginning
3. Insert at End
4. Insert at Position
5. Display
6. Exit
Enter your choice: 1
Enter number of nodes: 5
Enter data for node 1: 1
Enter data for node 2: 2
Enter data for node 3: 3
Enter data for node 4: 4
Enter data for node 5: 5
Linked list created successfully!
Enter your choice: 2
Enter data: 0
Node inserted at the beginning.
Enter your choice: 5
Linked list: 0
1
2
3
4
5
NULL
```

```
Enter your choice: 3
Enter data: 6
Node inserted at the end.
Enter your choice: 4
Enter data and position: 2
8
Node inserted at position 8.
Enter your choice: 5
Linked list: 0
1
2
3
4
5
6
2
NULL
Enter your choice: 6
```



```
printf("Enter data : ");
scanf("%d", &data);

newnode -> data = data;
newnode -> next = NULL;

if (head == NULL)
{
    head = newnode;
}
else {
    temp -> next = newnode;
}

}

b). void insertatBeginning()
{
    struct node, newnode*, temp*;
    int data;

    // create a new node.

    newnode=(struct node*)*malloc(sizeof(struct noo

    printf("Enter data : ");
    scanf("%d", &data);
```



```
Pg 4

a) Create a linked list
b) Insertion of a node at (i) first position
                          (ii) any position
                          (iii) end of the list
c) Display the contents of the list

a) struct node {
       int data;
       struct node * next;
   };

   struct Node *head = NULL;

   void createlist(int n)
   {
       struct node, newnode*, temp*;
       int data, i;

       if (n=0) {
           printf("Enter valid number of nodes : ");
       }
       else {
           for(i=0 ; i<=n ; i++)
           {
               newnode = (struct node)* malloc(sizeof
                                               (struct node));

               if (newnode = NULL);
               {
                   printf("memory allocation failed");
               }
           }
```

```
for(i=0; i<=pos-1; i++ && temp!=NULL)
{
    temp -> temp = next;
}

if (temp == null)
{
    printf(" out of range ");
    free(node);
}
else {
    newnode -> next = temp -> next;
    temp -> next = newnode;
}

}
```

⇒ at end.

```
void insertatend(int data)
{
    // create a node.
    newnode -> data = data;
    newnode -> next = NULL;

if (head = NULL)
{
    newnode -> data = head;
}
```

temp

```
newnode -> data = data;
newnode -> next = head;

}
```

⇒ Insert at any position.

```
void insertanyposition()
{
    struct node, newnode*, temp*;
    int data, pos;

// create a node
printf(" Enter data = ");
scanf("%d %d", &data, &pos);
for(int i=0; i<=pos-1; i++)
{
    temp
}
```

prev

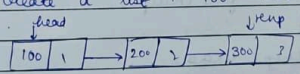
```
if (pos < 1)
{
    printf("Invalid position");
}

if (pos==1)
{
    insertatbeginning();
}

// create a node
newnode -> data = data;
```

c) Display

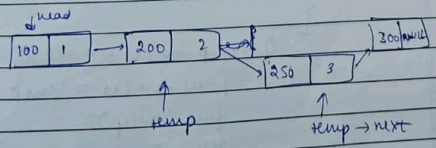
```
void display()
{
    if(head == NULL)
    {
        printf("List is empty.");
    }
    else
    {
        printf("Linked list :");
        while(temp != NULL)
        {
            printf("%d", temp->data);
        }
    }
```

8. Trace the following

a) create a list : 100 200 300



b) insert at pos 3 → 250



c) insert at end → 500