Page No.
Date / /

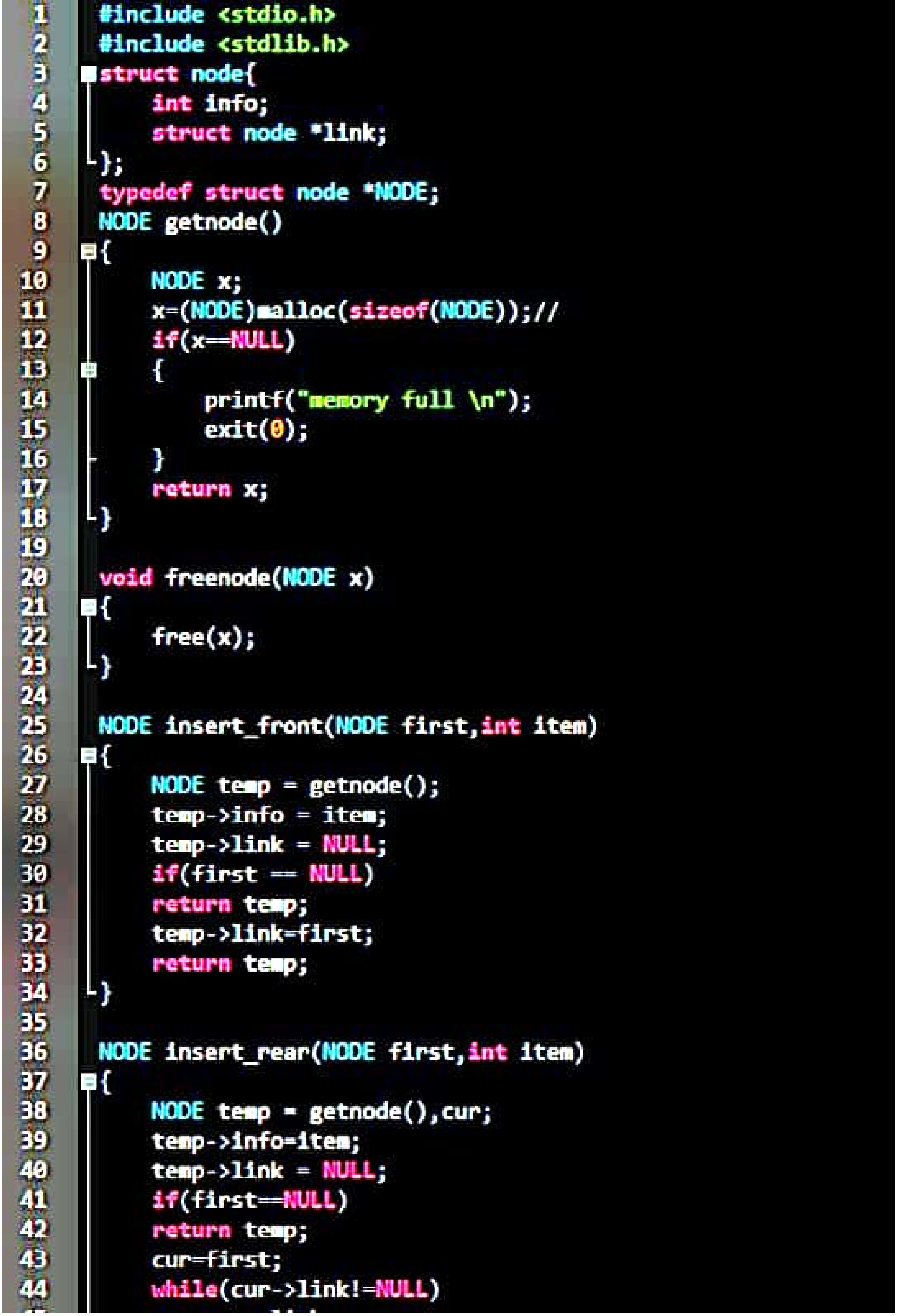
→ DS Lab Program 5

```
#include <stdio.h>
#include <stdlib.h>
struct node {
  int info;
  struct node *link;
};
typedef struct node NODE;
NODE getnode () {
  NODE x;
  x= (NODE)malloc (sizeof(NODE)).
  if (x = = NULL) {
     printf ("Memory full \n");
     exit (0);
  }
  return x;
}
void freenode (NODE x) {
   free(x);
}
NODE insert_front (NODE first, int item) {
 NODE temp = getnode ();
 temp -> info = item;
 temp -> link = NULL;
 if (first = = NULL)
     return temp;
 temp -> link = first;
 return temp;
 }


NODE insert_rear (NODE first, int item ) {
 NODE temp = getnode (), cur;
 temp -> info = item;
temp -> link = NULL;
if (first == NULL)
    return temp;
cur = first
while (cur ->link != NULL)
    cur = cur -> link;
cur -> link = temp;
return first;
}
void insert_at (NODE first int item) {
  printf("Enter position after which to enter\n");
  int pos;
  scanf("%d", &pos);
  NODE bef = getnode ();
  bef = first;
  NODE nxt = getnode ();
  NODE temp = get node ();
  temp -> info = item;
  temp -> link = NULL;
  for (int i = 1; i < pos; i++)
      bef = bef -> link;
  ~~nd~~ nxt = bef -> link;
  bef -> link = temp;
  temp -> link = nxt;
} ~~printf~~
void display (NODE first) {
  if (first == NULL) {
      printf("List is empty \n");
      return;
  }
  printf("Elements of list : \n");
  for (NODE i = first; i != NULL; i = i -> link)
    printf("%d\n", i -> info);
}
```

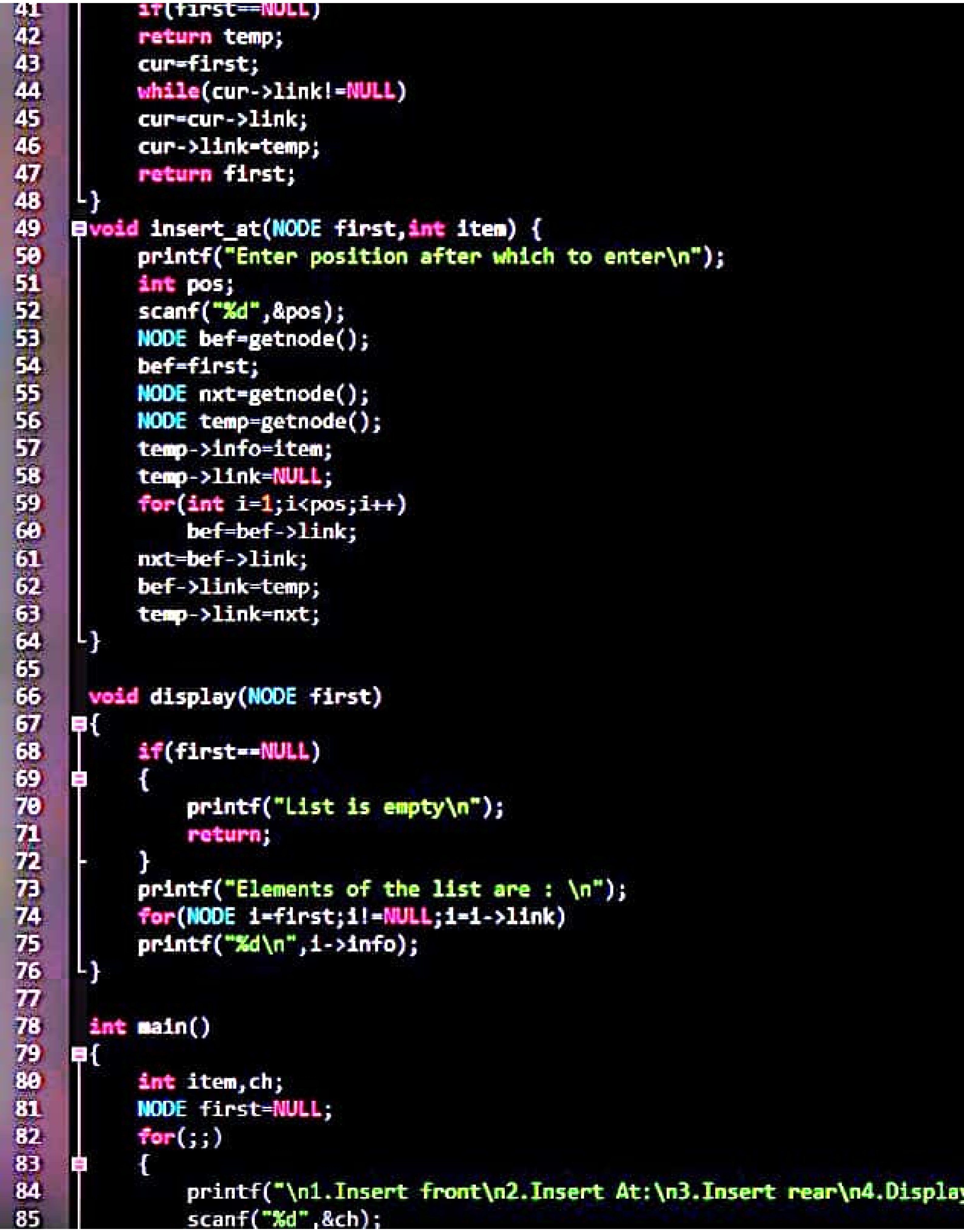
Date / /

```
int main () {
int item, ch;
NODE first = NULL;
for (;;) {
printf ("\n 1. Insert front \n 2. Insert At \n 3.
        Insert rear \n 4. Display \n");
scanf ("%d", &ch);
switch (ch) {
  case 1: printf ("Enter element to be inserted \n");
          scanf ("%d", &item);
          first = insert_front (first, item);
          break;
  case 2: printf ("Enter element to insert \n");
          scanf ("%d", &item);
          insert_at (first, item);
          break;
  case 3: printf ("Enter element to insert \n");
          scanf ("%d", &item);
          first = insert_rear (first, item);
          break;
  case 4: display (first);
          break;
  default: return 0;
  }
 }
}
```

Scanned with CamScanner

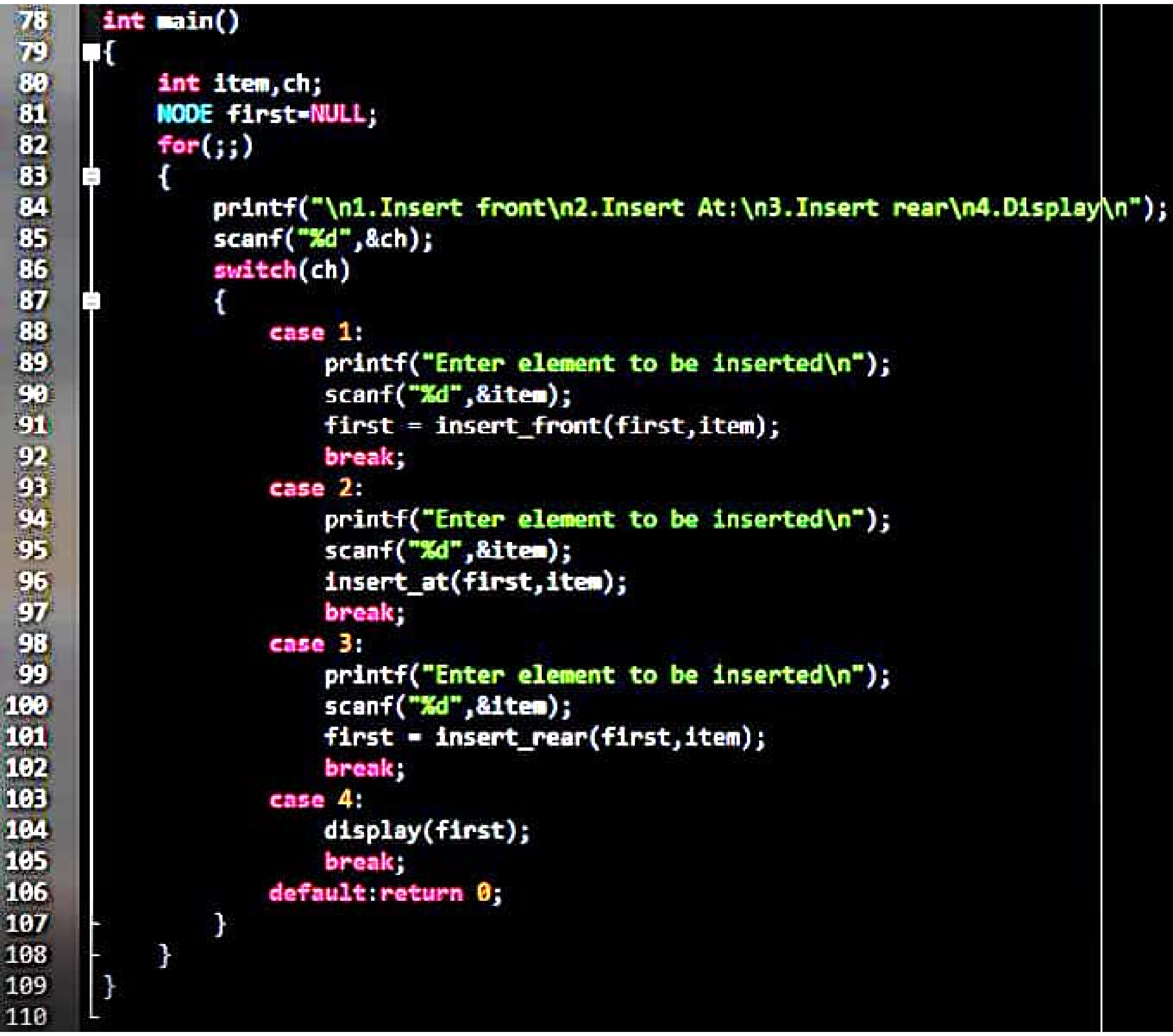


```c
#include <stdio.h>
#include <stdlib.h>
struct node{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x=(NODE)malloc(sizeof(NODE));//
    if(x==NULL)
    {
        printf("memory full \n");
        exit(0);
    }
    return x;
}

void freenode(NODE x)
{
    free(x);
}

NODE insert_front(NODE first,int item)
{
    NODE temp = getnode();
    temp->info = item;
    temp->link = NULL;
    if(first == NULL)
    return temp;
    temp->link=first;
    return temp;
}

NODE insert_rear(NODE first,int item)
{
    NODE temp = getnode(),cur;
    temp->info=item;
    temp->link = NULL;
    if(first==NULL)
    return temp;
    cur=first;
    while(cur->link!=NULL)
```

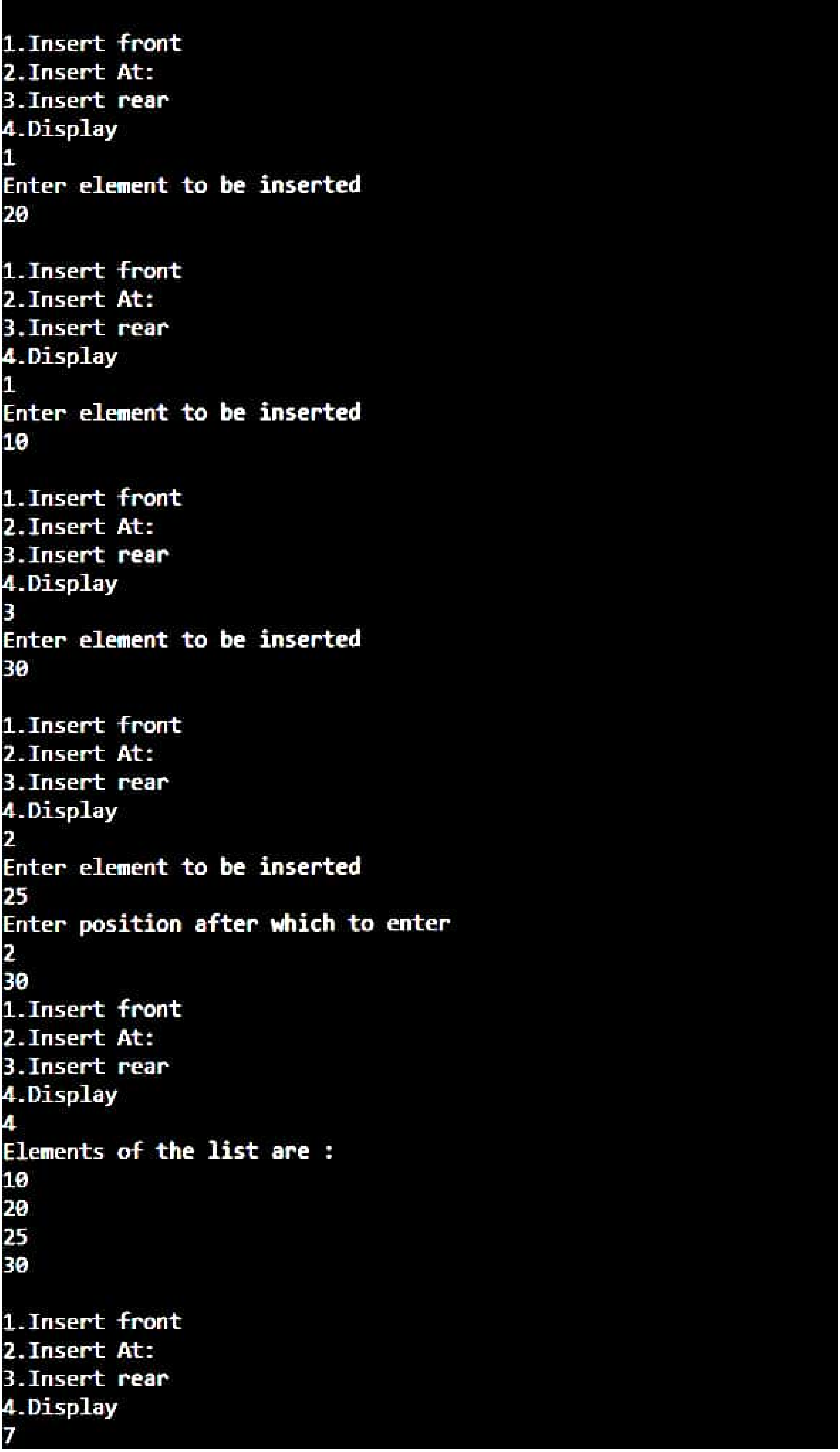


```c
    if(first==NULL)
    return temp;
    cur=first;
    while(cur->link!=NULL)
    cur=cur->link;
    cur->link=temp;
    return first;
}
void insert_at(NODE first,int item) {
    printf("Enter position after which to enter\n");
    int pos;
    scanf("%d",&pos);
    NODE bef=getnode();
    bef=first;
    NODE nxt=getnode();
    NODE temp=getnode();
    temp->info=item;
    temp->link=NULL;
    for(int i=1;i<pos;i++)
        bef=bef->link;
    nxt=bef->link;
    bef->link=temp;
    temp->link=nxt;
}

void display(NODE first)
{
    if(first==NULL)
    {
        printf("List is empty\n");
        return;
    }
    printf("Elements of the list are : \n");
    for(NODE i=first;i!=NULL;i=i->link)
    printf("%d\n",i->info);
}

int main()
{
    int item,ch;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1.Insert front\n2.Insert At:\n3.Insert rear\n4.Display
        scanf("%d",&ch);
```

Scanned with CamScanner



```c
int main()
{
    int item,ch;
    NODE first=NULL;
    for(;;)
    {
        printf("\n1.Insert front\n2.Insert At:\n3.Insert rear\n4.Display\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("Enter element to be inserted\n");
                scanf("%d",&item);
                first = insert_front(first,item);
                break;
            case 2:
                printf("Enter element to be inserted\n");
                scanf("%d",&item);
                insert_at(first,item);
                break;
            case 3:
                printf("Enter element to be inserted\n");
                scanf("%d",&item);
                first = insert_rear(first,item);
                break;
            case 4:
                display(first);
                break;
            default:return 0;
        }
    }
}
```



```
1.Insert front
2.Insert At:
3.Insert rear
4.Display
1
Enter element to be inserted
20

1.Insert front
2.Insert At:
3.Insert rear
4.Display
1
Enter element to be inserted
10

1.Insert front
2.Insert At:
3.Insert rear
4.Display
3
Enter element to be inserted
30

1.Insert front
2.Insert At:
3.Insert rear
4.Display
2
Enter element to be inserted
25
Enter position after which to enter
2
30
1.Insert front
2.Insert At:
3.Insert rear
4.Display
4
Elements of the list are :
10
20
25
30

1.Insert front
2.Insert At:
3.Insert rear
4.Display
7
```