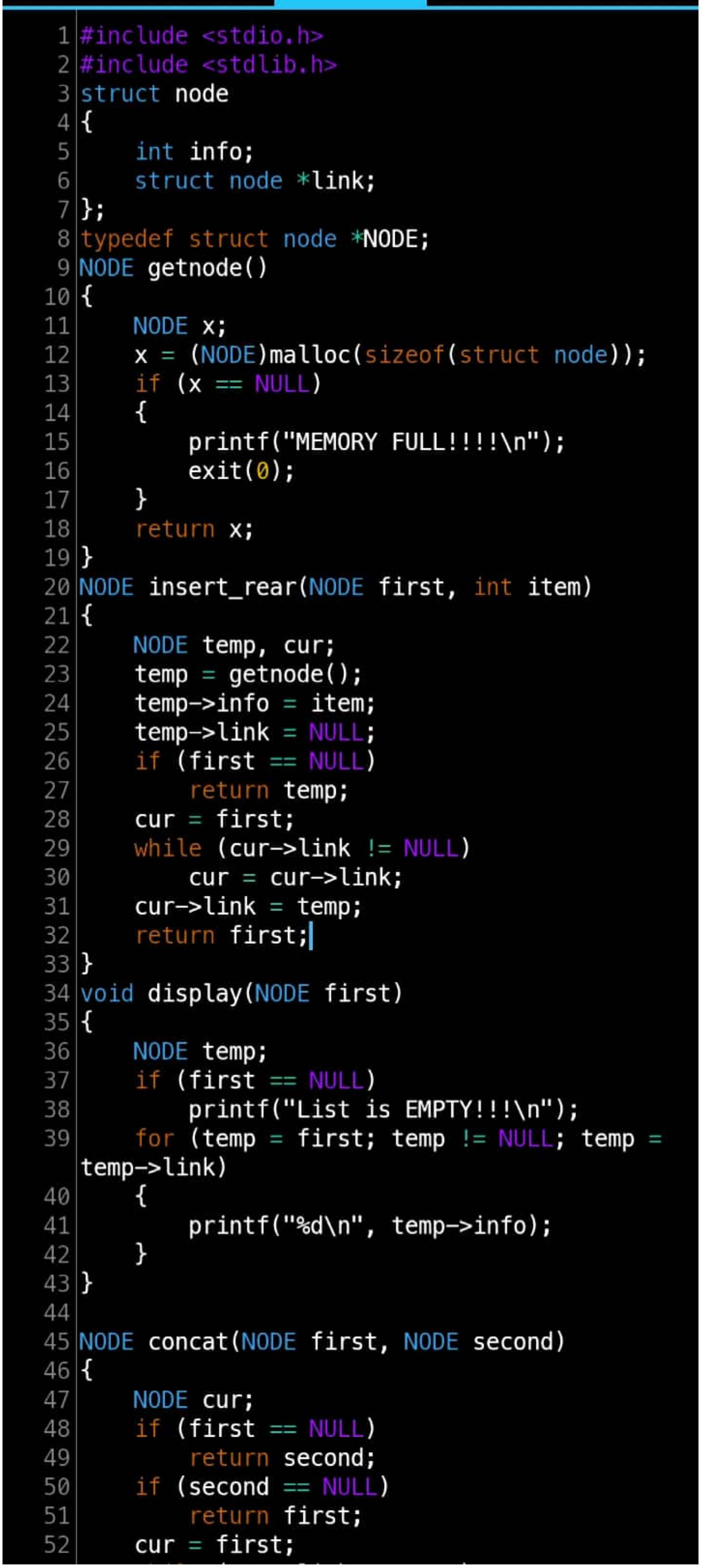
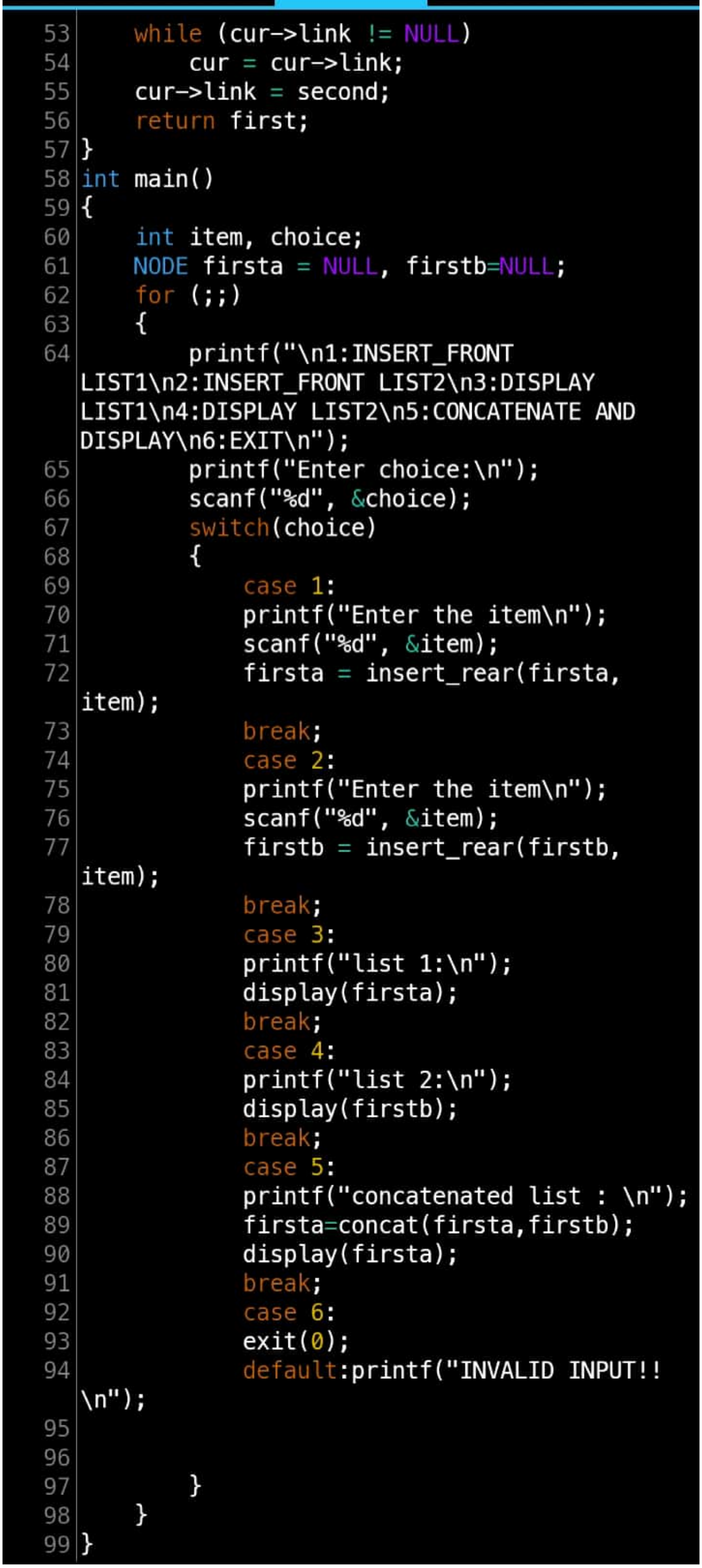
```
# include <stdio.h>
struct node
 {
  int info;
  struct node *link;
 }
typedef struct node *NODE;
  NODE getnode()
  {
   NODE x;
   x = (NODE) malloc(sizeof(struct node));
   if (x == NULL)
   {
    printf ("Memory Full !!! \n");
    exit (0);
   }
   return x;
  }
void freenode (NODE x)
 {  free(x)  }
NODE insert (NODE first, int item)
{
 NODE temp = getnode(), cur, prev;
 temp -> info = item;
 if (first == NULL)
 {
   first = getnode();
   first -> info = item;
return first;
}
if (item < first -> info)
  {
   temp -> link = first;
   return temp;
  }
  cur = first;
  prev = NULL;
  while ( cur != NULL && item > cur -> info)
   {
    prev = cur;
    cur = cur -> link.
   }
   prev -> link = temp;
   temp -> link = cur;
   return first;
   }
NODE reverse_list (NODE first)
{
 NODE cur, temp;
 Cur = NULL;
 while (first != NULL)
  {
   temp = first;
   first = first -> link;
   temp -> link = cur;
   cur = temp;
  }
 printf ("list has been reversed succesfully \n");
  return cur;
 }
```

Scanned with CamScanner

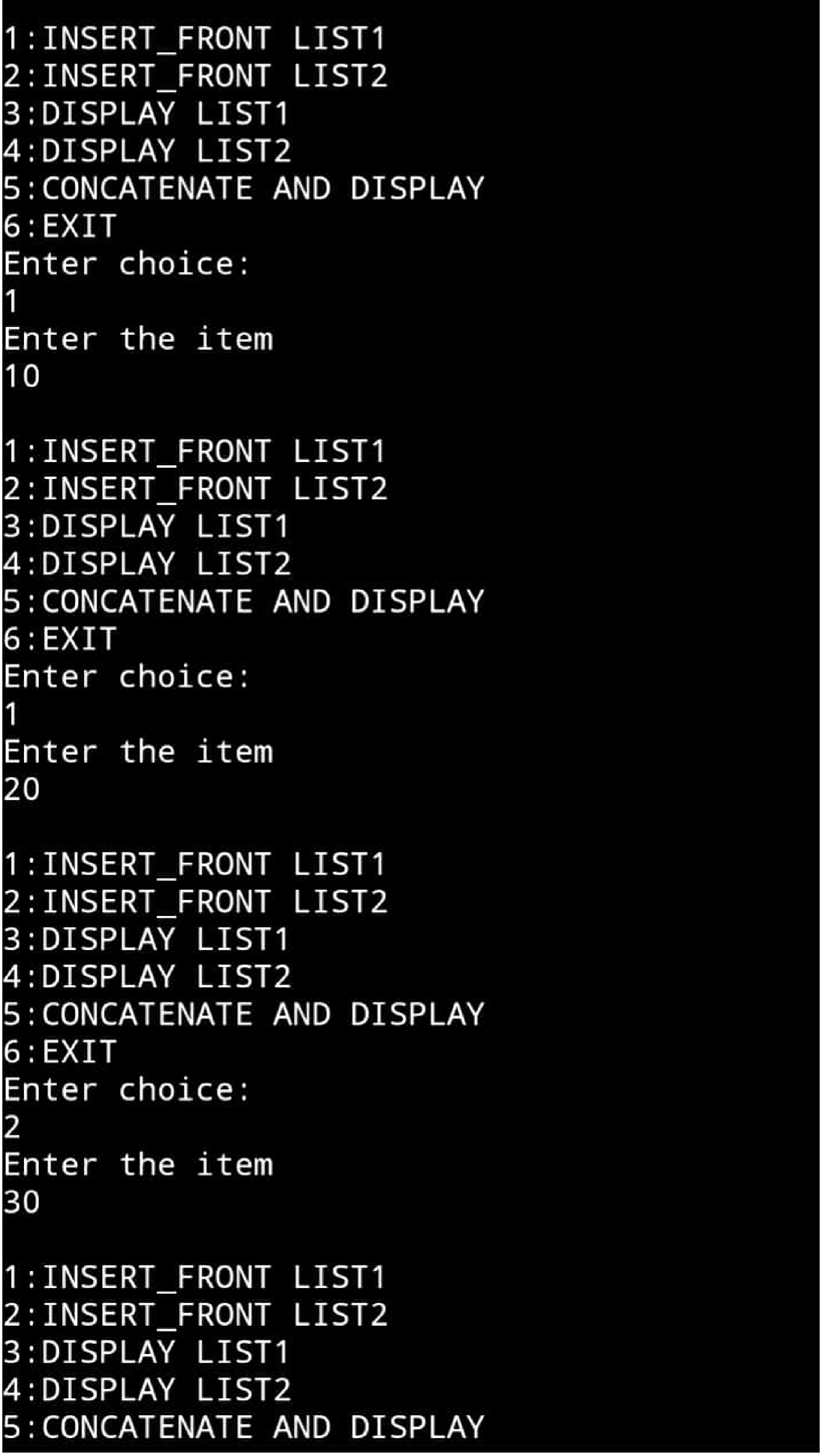
```
NODE concat (NODE first, NODE second)
{
 NODE cur;
 if (first == NULL)
   return second;
 if (second == NULL)
   return first;
  cur = first;
  while (cur->link != NULL)
   cur = cur->link;
   cur->link = second;
   return first;
}
void display (NODE first)
{
  NODE temp;
  if (first == NULL)
  printf ("List is EMPTY!!!\n");
   for (temp = first; temp != NULL; temp = temp->link)
    {
      printf ("%d \n", temp->info);
    }
}
void main ()
{
 int item, choice, pos, i, n;
 NODE firsta = NULL, firstb = NULL;
 for(;;)
  {
    printf ("\n 1. Insert List 1 \n 2. Insert List2 \n
              3. Reverse List1 \n 4. Reverse List2 \n 5. Display
             List1 \n 6. Display List2 \n 7. Concatenate \n
*   8. Exit \n");
printf ("Enter choice: \n");
 scanf ("%d", &choice);
 switch (Choice)
 {
  case 1 : printf ("Enter the element to be inserted \n");
           scanf ("%d", & item);
           ~~switch (ch)~~
           firsta = insert (firsta, item);
           break;
  case 2 : printf ("Enter the element to be inserted \n");
           scanf ("%d", &item);
           firstb = insert (firstb, item);
           break;
   case 3 : first a = reverse_list (firsta);
            break;
    case 4 : first b = reverse_list (firstb);
             break;
    case 5 : printf ("List 1 : \n");
             display (firsta);
             break;
    case 6 : printf ("List 2 : \n");
             display (firstb);
             break;
     case 7 : printf ("Concatenated list : \n");
              firsta = Concat (firsta, firstb);
              display (firsta);
              break;
     case 8 : exit (0);
     default : printf ("Invalid input !! \n");
     }
     } }
```

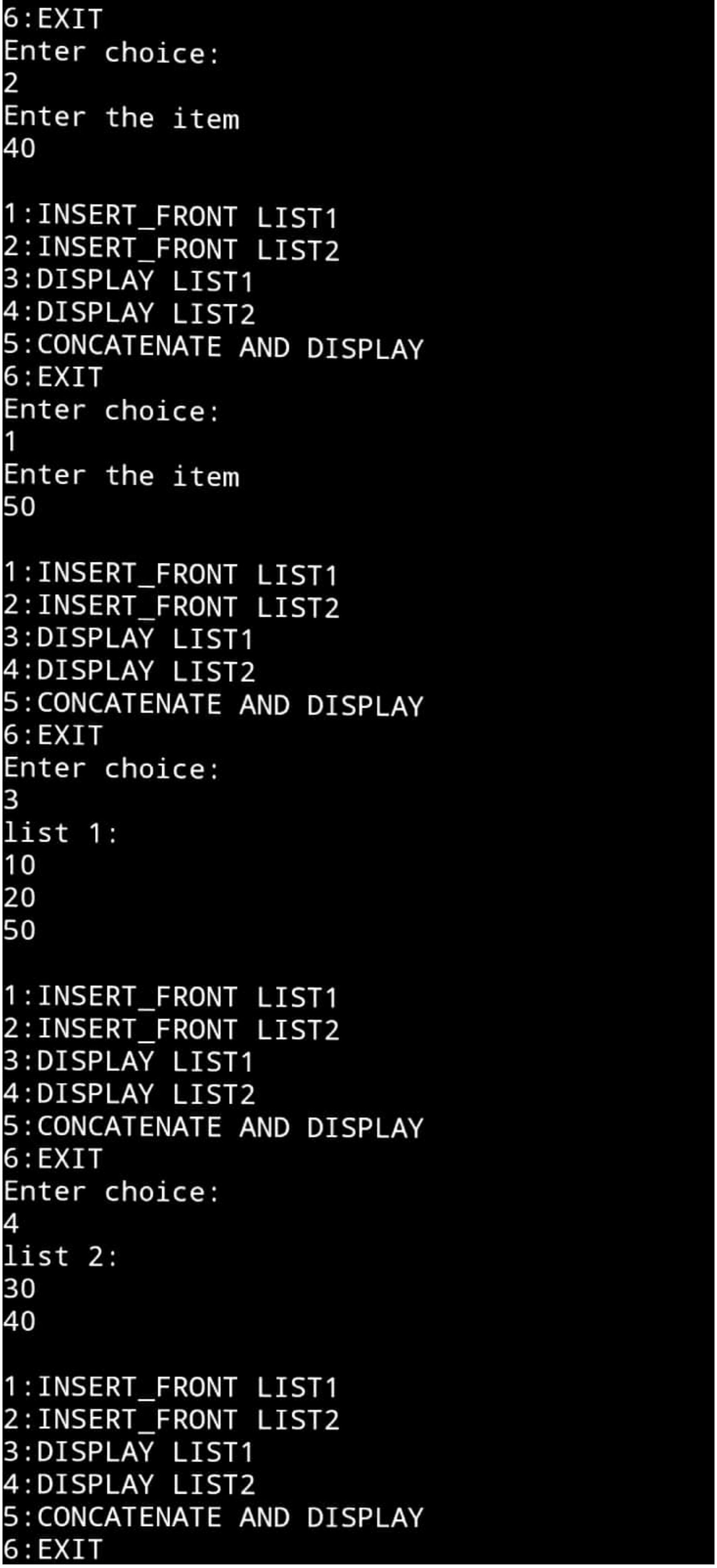


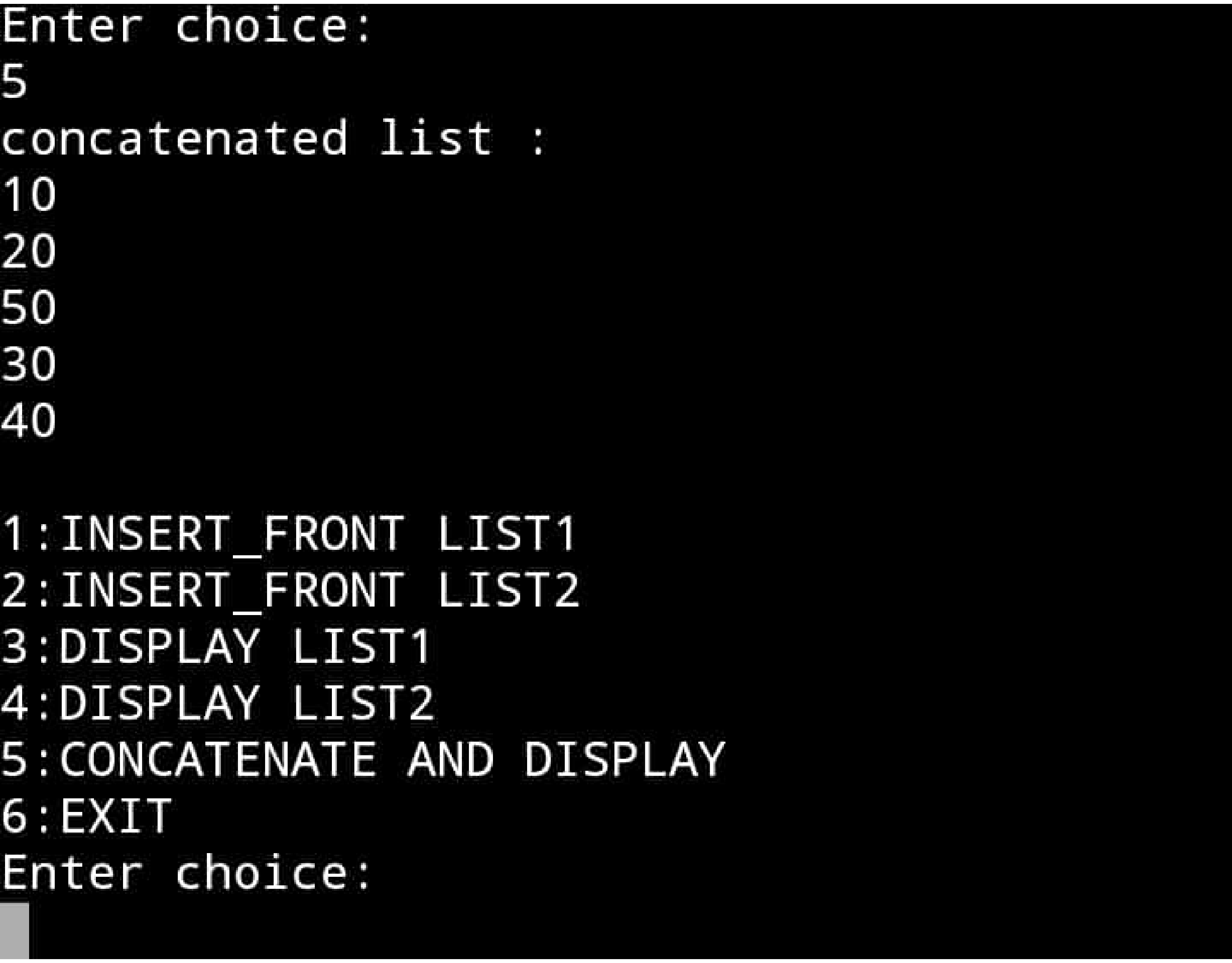
```c
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int info;
    struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE)malloc(sizeof(struct node));
    if (x == NULL)
    {
        printf("MEMORY FULL!!!!\n");
        exit(0);
    }
    return x;
}
NODE insert_rear(NODE first, int item)
{
    NODE temp, cur;
    temp = getnode();
    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 display(NODE first)
{
    NODE temp;
    if (first == NULL)
        printf("List is EMPTY!!!\n");
    for (temp = first; temp != NULL; temp = temp->link)
    {
        printf("%d\n", temp->info);
    }
}

NODE concat(NODE first, NODE second)
{
    NODE cur;
    if (first == NULL)
        return second;
    if (second == NULL)
        return first;
    cur = first;
```



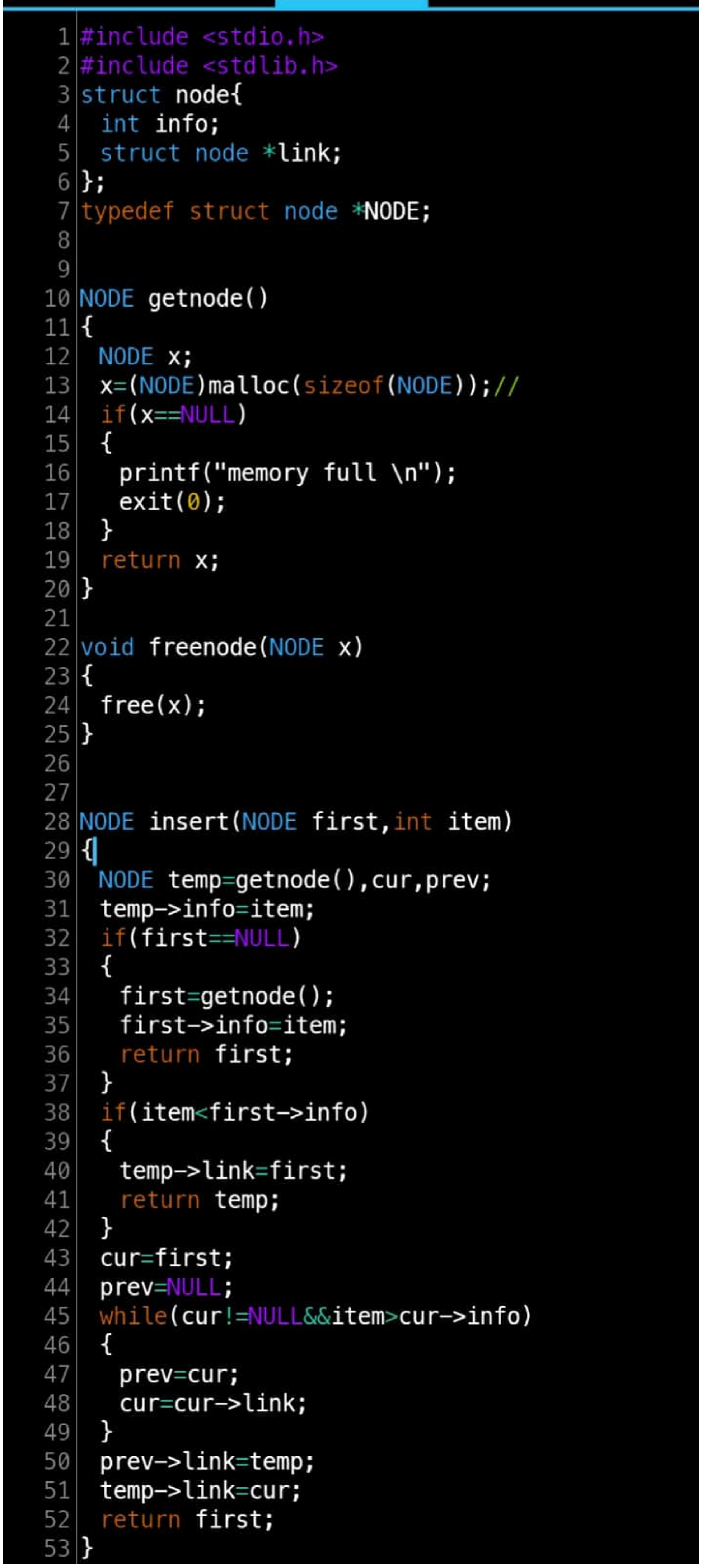
```c
    while (cur->link != NULL)
        cur = cur->link;
    cur->link = second;
    return first;
}
int main()
{
    int item, choice;
    NODE firsta = NULL, firstb=NULL;
    for (;;)
    {
        printf("\n1:INSERT_FRONT LIST1\n2:INSERT_FRONT LIST2\n3:DISPLAY LIST1\n4:DISPLAY LIST2\n5:CONCATENATE AND DISPLAY\n6:EXIT\n");
        printf("Enter choice:\n");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
            printf("Enter the item\n");
            scanf("%d", &item);
            firsta = insert_rear(firsta, item);
            break;
            case 2:
            printf("Enter the item\n");
            scanf("%d", &item);
            firstb = insert_rear(firstb, item);
            break;
            case 3:
            printf("list 1:\n");
            display(firsta);
            break;
            case 4:
            printf("list 2:\n");
            display(firstb);
            break;
            case 5:
            printf("concatenated list : \n");
            firsta=concat(firsta,firstb);
            display(firsta);
            break;
            case 6:
            exit(0);
            default:printf("INVALID INPUT!! \n");


        }
    }
}
```


```
1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
1
Enter the item
10

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
1
Enter the item
20

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
2
Enter the item
30

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
```

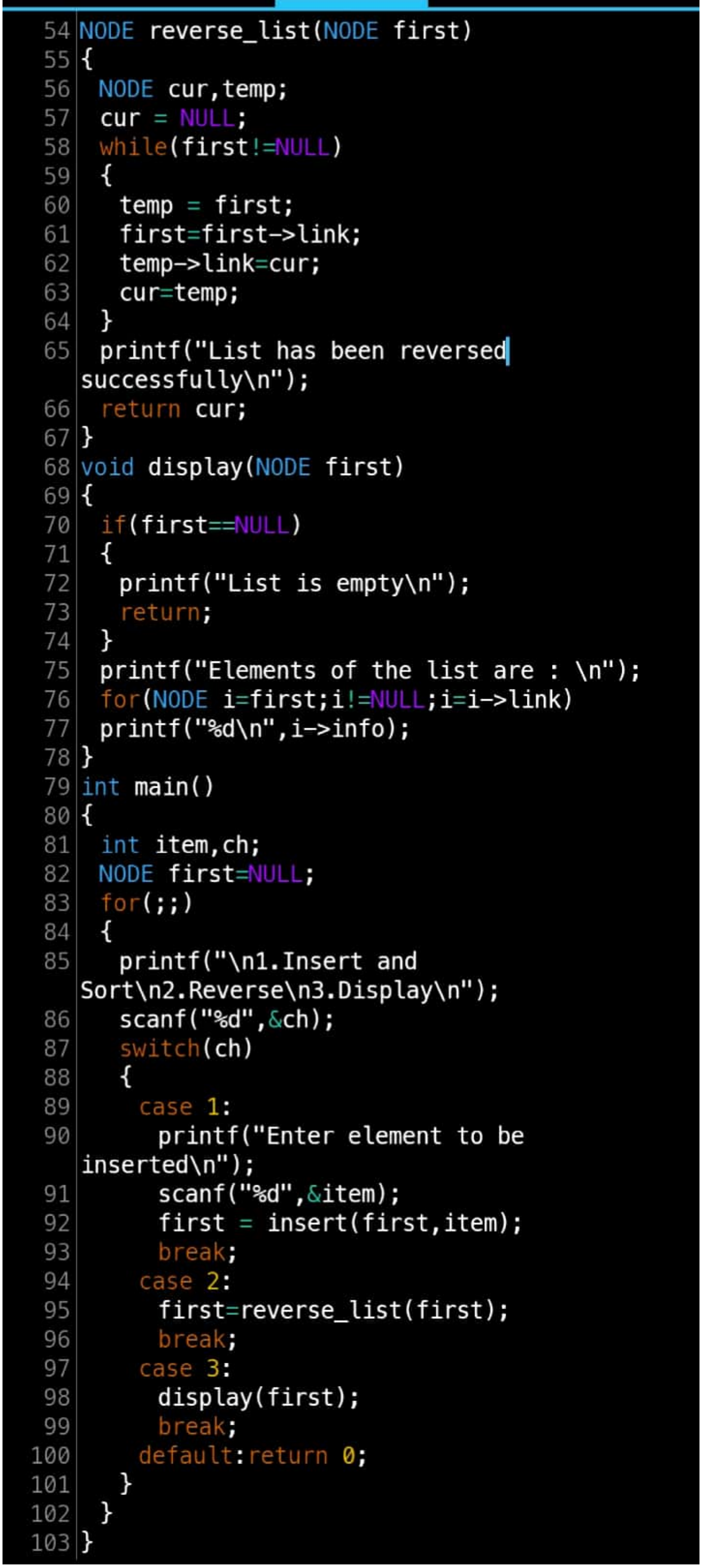

```
6:EXIT
Enter choice:
2
Enter the item
40

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
1
Enter the item
50

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
3
list 1:
10
20
50

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
4
list 2:
30
40

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
```



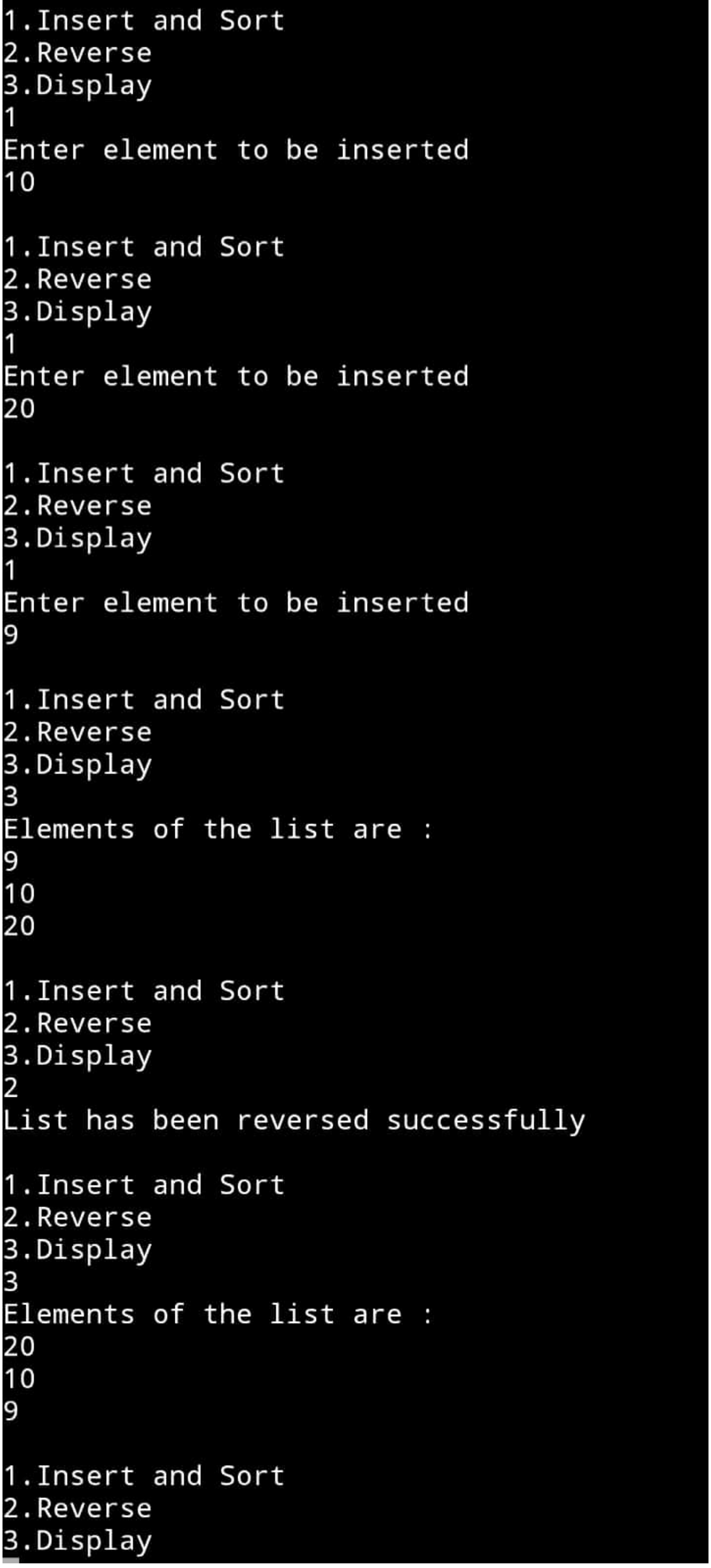
```
Enter choice:
5
concatenated list :
10
20
50
30
40

1:INSERT_FRONT LIST1
2:INSERT_FRONT LIST2
3:DISPLAY LIST1
4:DISPLAY LIST2
5:CONCATENATE AND DISPLAY
6:EXIT
Enter choice:
```



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



```c
NODE reverse_list(NODE first)
{
 NODE cur,temp;
 cur = NULL;
 while(first!=NULL)
 {
  temp = first;
  first=first->link;
  temp->link=cur;
  cur=temp;
 }
 printf("List has been reversed successfully\n");
 return cur;
}
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 and Sort\n2.Reverse\n3.Display\n");
  scanf("%d",&ch);
  switch(ch)
  {
   case 1:
    printf("Enter element to be inserted\n");
    scanf("%d",&item);
    first = insert(first,item);
    break;
   case 2:
    first=reverse_list(first);
    break;
   case 3:
    display(first);
    break;
   default:return 0;
  }
 }
}
```



```
1.Insert and Sort
2.Reverse
3.Display
1
Enter element to be inserted
10

1.Insert and Sort
2.Reverse
3.Display
1
Enter element to be inserted
20

1.Insert and Sort
2.Reverse
3.Display
1
Enter element to be inserted
9

1.Insert and Sort
2.Reverse
3.Display
3
Elements of the list are :
9
10
20

1.Insert and Sort
2.Reverse
3.Display
2
List has been reversed successfully

1.Insert and Sort
2.Reverse
3.Display
3
Elements of the list are :
20
10
9

1.Insert and Sort
2.Reverse
3.Display
```