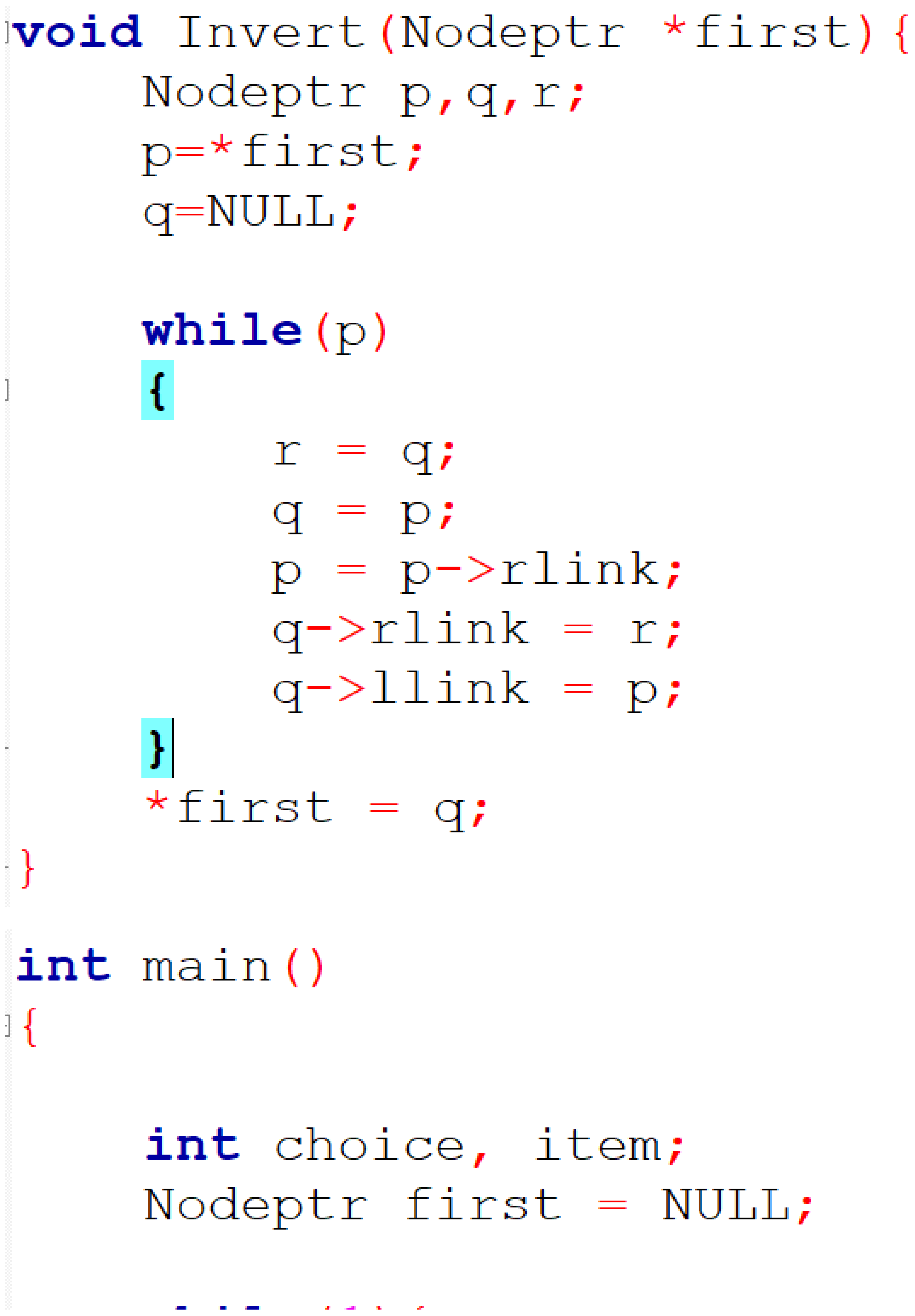
```c
// Doubly linked lists
typedef struct node *Nodeptr;

typedef struct node
{
    int data;
    Nodeptr llink;
    Nodeptr rlink;
}NODE;

Nodeptr getnode()
{
    Nodeptr temp;
    temp = (Nodeptr) malloc(sizeof(NODE));
    return temp;
}

int IsEmpty(Nodeptr first){ return (first==NULL) ?1:0;
void Display(Nodeptr first)
{
    Nodeptr temp;

    if (IsEmpty(first))
    {
        printf("Empty List"); return;
    }

    printf("\nContents of List : ");
    for(temp=first;temp;temp=temp->rlink)
        printf("%d ",temp->data);
    printf("\n");
}
```

```c
void InsertRear( int item, Nodeptr *first)
{
    Nodeptr temp,last;
    temp = getnode();
    temp->data = item;
    temp->llink = NULL;
    temp->rlink = NULL;

    if (IsEmpty(*first)){ *first = temp; return;   }

    last = *first;
    while(last->rlink)
        last = last->rlink;
    last->rlink = temp;
    temp->llink = last;
}
```

```c
void InsertFront(int item, Nodeptr *first)
{
    Nodeptr temp;

    temp = getnode();
    temp->data = item;
    temp->llink = NULL;
    temp->rlink = *first;

    if (!IsEmpty(*first))
        (*first)->llink = temp;
    *first = temp;

}
```

```
int  DeleteFront(Nodeptr *first)
{
    Nodeptr temp;
    int num;
    if (IsEmpty(*first)){
            printf("Empty List\n");return ERROR;
    }

    num = (*first)->data;
    temp = *first;
    *first = (*first)->rlink;

    if(*first) //if not deleting the last remaining node
        (*first)->llink = NULL;
    free(temp);
    return(num);
}

int DeleteRear(Nodeptr *first){
    Nodeptr prev,cur;
    int num;

    if (IsEmpty(*first)){
            printf("Empty List\n");return ERROR; }
    cur = *first;
    while(cur->rlink)
        cur = cur->rlink;
    prev = cur->llink;
    if (prev)  prev->rlink = NULL;
    else
        *first = NULL;//deleting the last remaining node
    num = cur->data;
    free(cur);
    return(num);
}
```

```c
Nodeptr Search(int item, Nodeptr first){
    Nodeptr cur;

    if (IsEmpty(first)){
        printf("Empty List\n"); return NULL;
    }

    cur = first;
    while(cur){
        if (cur->data ==item)
            return cur;
        cur = cur->rlink;
    }
    return NULL;
}
```



```
void Invert(Nodeptr *first){
    Nodeptr p,q,r;
    p=*first;
    q=NULL;

    while(p)
    {
        r = q;
        q = p;
        p = p->rlink;
        q->rlink = r;
        q->llink = p;
    }
    *first = q;
}

int main()
{

    int choice, item;
    Nodeptr first = NULL;

```

}