# Sardar Vallabhbhai National Institute of Technology, Surat
## Department of Artificial Intelligence

**Data Structure (AI102)**

**B.Tech I - II Semester**

**<u>Assignment-5</u>**

Note: Input should be taken from the user

**Q1: Write a C/C++ program to implement a circular linked list with the following operations:**

a) Insert an element at a specific position specified by the user.

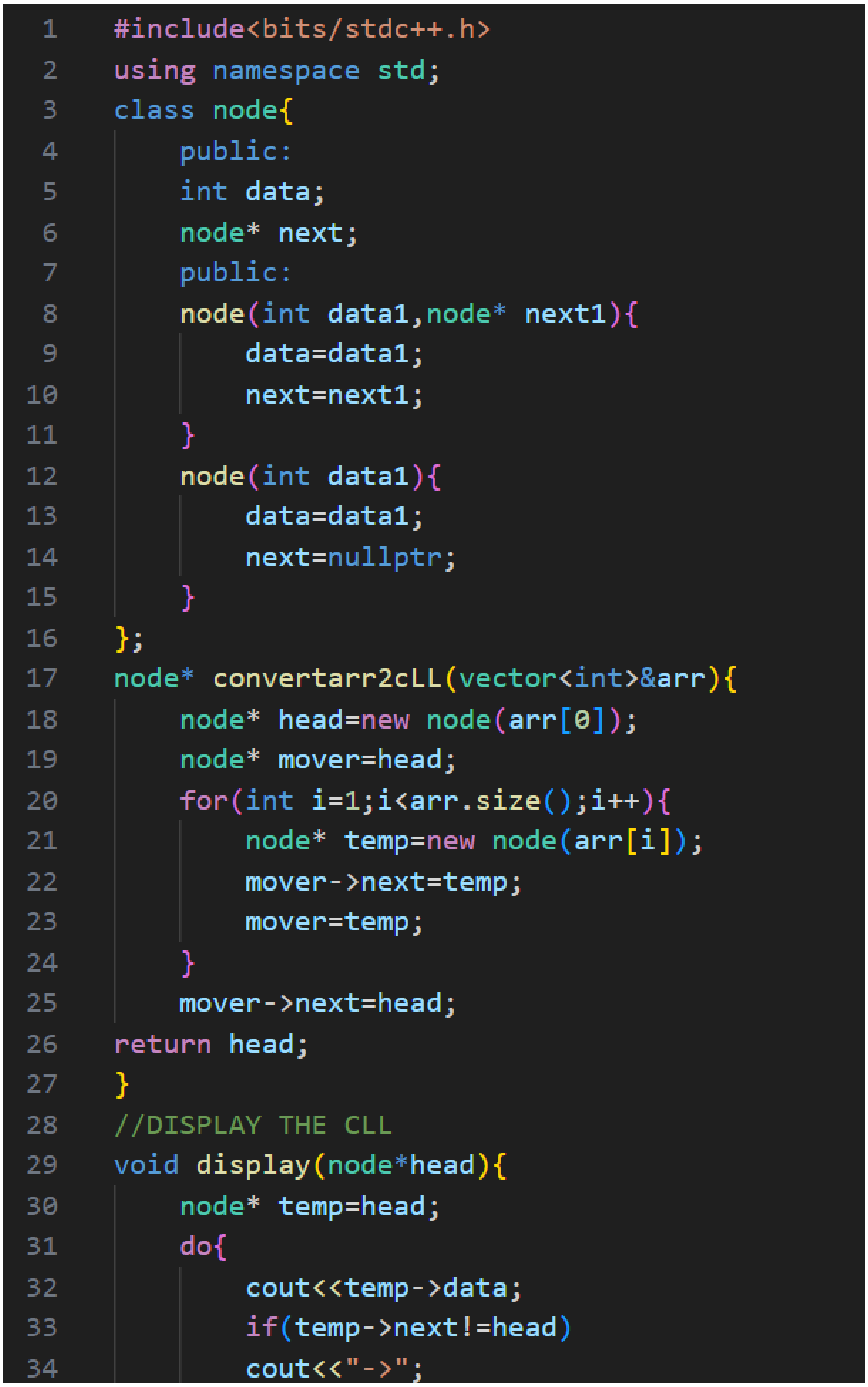
b) Insert an element at the beginning of the list

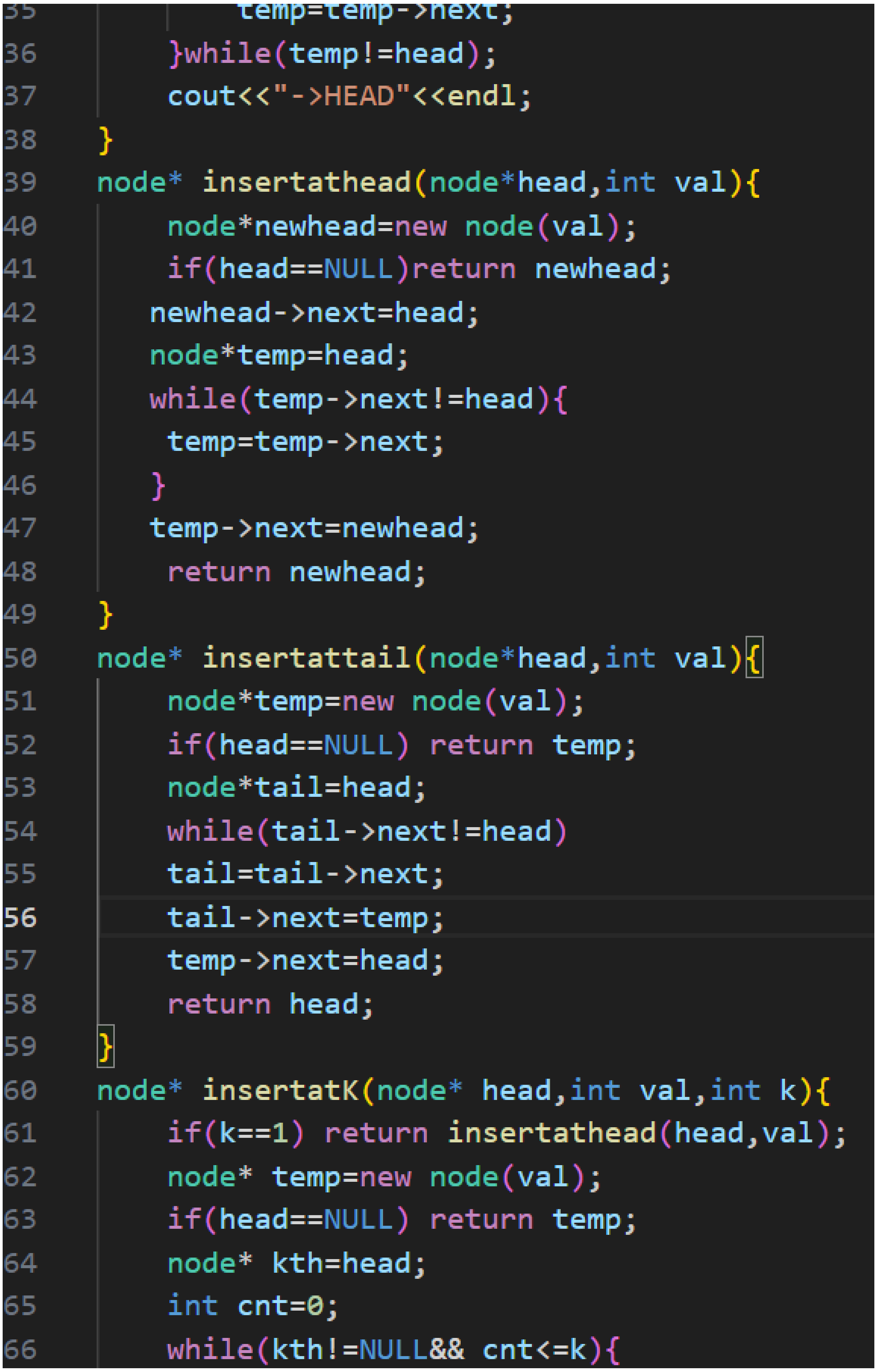
c) Insert an element at the end of the list.

d) Delete an element from a specific position specified by the user.

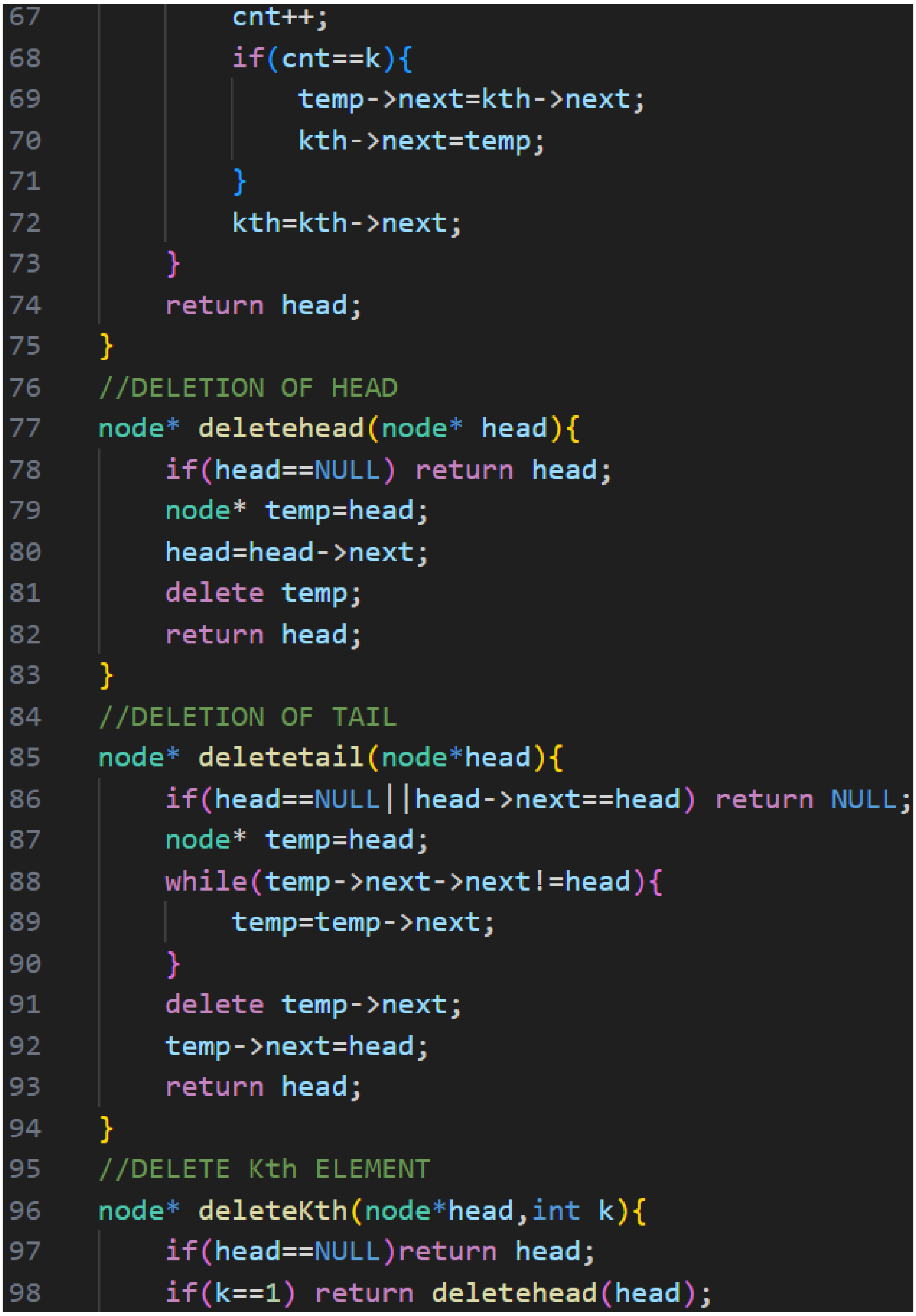
e) Delete the first element from the list.

f) Delete the last element from the list.

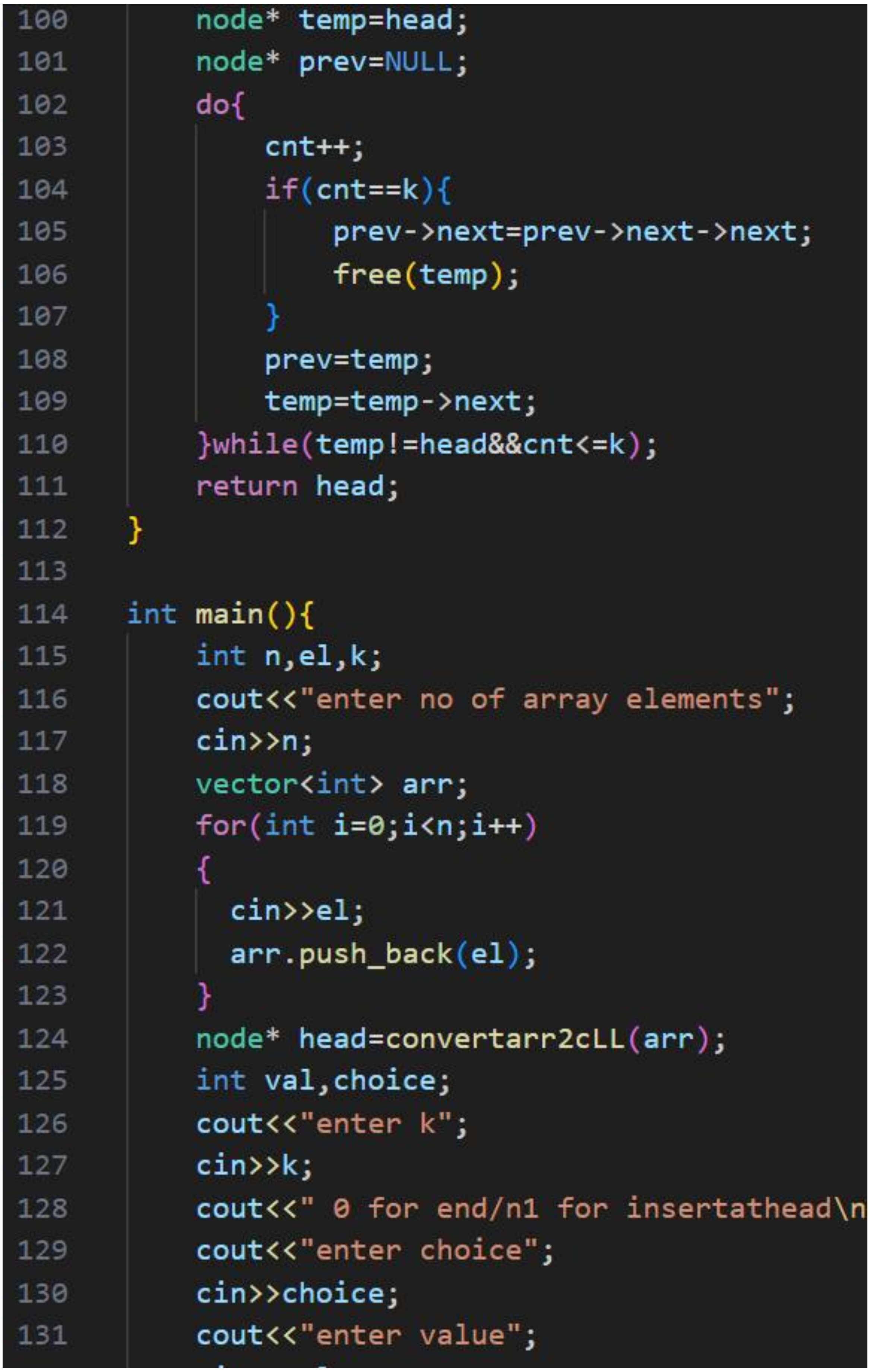

```cpp
#include<bits/stdc++.h>
using namespace std;
class node{
    public:
    int data;
    node* next;
    public:
    node(int data1,node* next1){
        data=data1;
        next=next1;
    }
    node(int data1){
        data=data1;
        next=nullptr;
    }
};
node* convertarr2cLL(vector<int>&arr){
    node* head=new node(arr[0]);
    node* mover=head;
    for(int i=1;i<arr.size();i++){
        node* temp=new node(arr[i]);
        mover->next=temp;
        mover=temp;
    }
    mover->next=head;
return head;
}
//DISPLAY THE CLL
void display(node*head){
    node* temp=head;
    do{
        cout<<temp->data;
        if(temp->next!=head)
        cout<<"->";
```



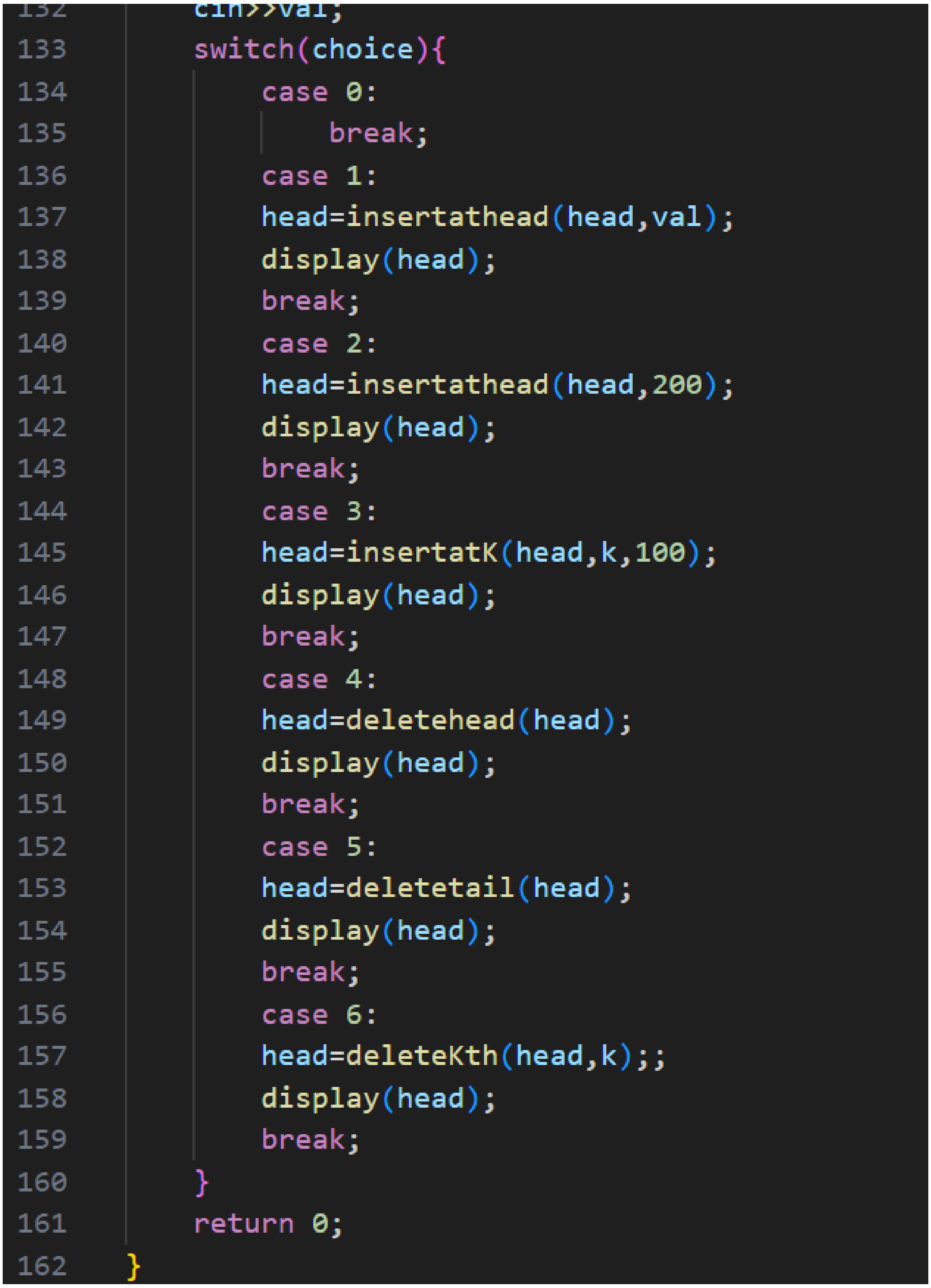
```
        temp=temp->next;
    }while(temp!=head);
    cout<<"->HEAD"<<endl;
}
node* insertathead(node*head,int val){
    node*newhead=new node(val);
    if(head==NULL)return newhead;
   newhead->next=head;
   node*temp=head;
   while(temp->next!=head){
    temp=temp->next;
   }
   temp->next=newhead;
    return newhead;
}
node* insertattail(node*head,int val){
    node*temp=new node(val);
    if(head==NULL) return temp;
    node*tail=head;
    while(tail->next!=head)
    tail=tail->next;
    tail->next=temp;
    temp->next=head;
    return head;
}
node* insertatK(node* head,int val,int k){
    if(k==1) return insertathead(head,val);
    node* temp=new node(val);
    if(head==NULL) return temp;
    node* kth=head;
    int cnt=0;
    while(kth!=NULL&& cnt<=k){
```



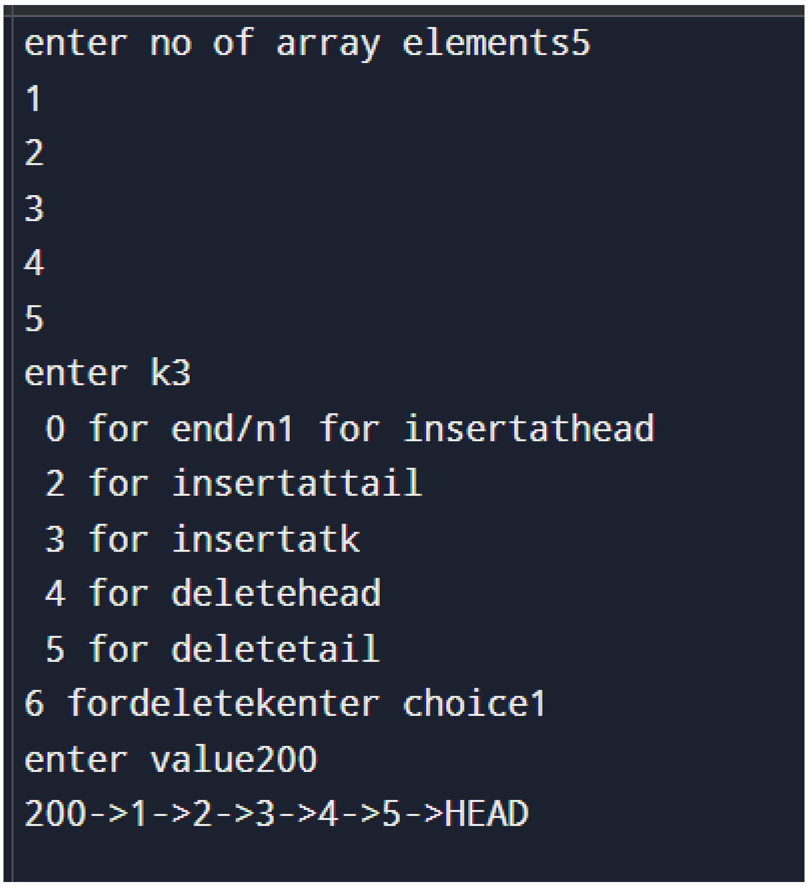
```
            cnt++;
            if(cnt==k){
                temp->next=kth->next;
                kth->next=temp;
            }
            kth=kth->next;
        }
        return head;
    }
    //DELETION OF HEAD
    node* deletehead(node* head){
        if(head==NULL) return head;
        node* temp=head;
        head=head->next;
        delete temp;
        return head;
    }
    //DELETION OF TAIL
    node* deletetail(node*head){
        if(head==NULL||head->next==head) return NULL;
        node* temp=head;
        while(temp->next->next!=head){
            temp=temp->next;
        }
        delete temp->next;
        temp->next=head;
        return head;
    }
    //DELETE Kth ELEMENT
    node* deleteKth(node*head,int k){
        if(head==NULL)return head;
        if(k==1) return deletehead(head);
```



```cpp
    node* temp=head;
    node* prev=NULL;
    do{
        cnt++;
        if(cnt==k){
            prev->next=prev->next->next;
            free(temp);
        }
        prev=temp;
        temp=temp->next;
    }while(temp!=head&&cnt<=k);
    return head;
}

int main(){
    int n,el,k;
    cout<<"enter no of array elements";
    cin>>n;
    vector<int> arr;
    for(int i=0;i<n;i++)
    {
      cin>>el;
      arr.push_back(el);
    }
    node* head=convertarr2cLL(arr);
    int val,choice;
    cout<<"enter k";
    cin>>k;
    cout<<" 0 for end/n1 for insertathead\n
    cout<<"enter choice";
    cin>>choice;
    cout<<"enter value";
```



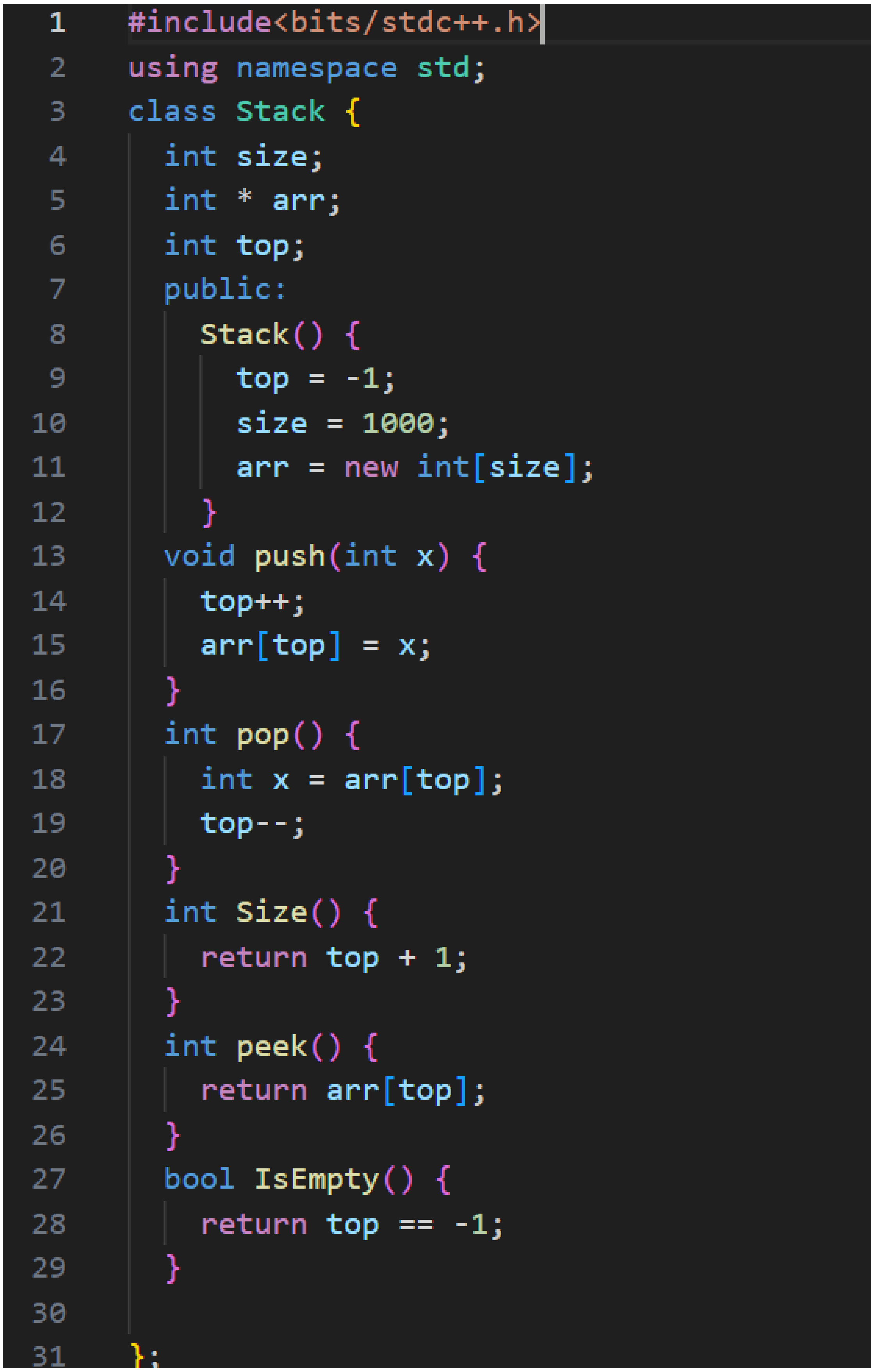
```
    cin>>val;
    switch(choice){
        case 0:
            break;
        case 1:
        head=insertathead(head,val);
        display(head);
        break;
        case 2:
        head=insertathead(head,200);
        display(head);
        break;
        case 3:
        head=insertatK(head,k,100);
        display(head);
        break;
        case 4:
        head=deletehead(head);
        display(head);
        break;
        case 5:
        head=deletetail(head);
        display(head);
        break;
        case 6:
        head=deleteKth(head,k);;
        display(head);
        break;
    }
    return 0;
}
```



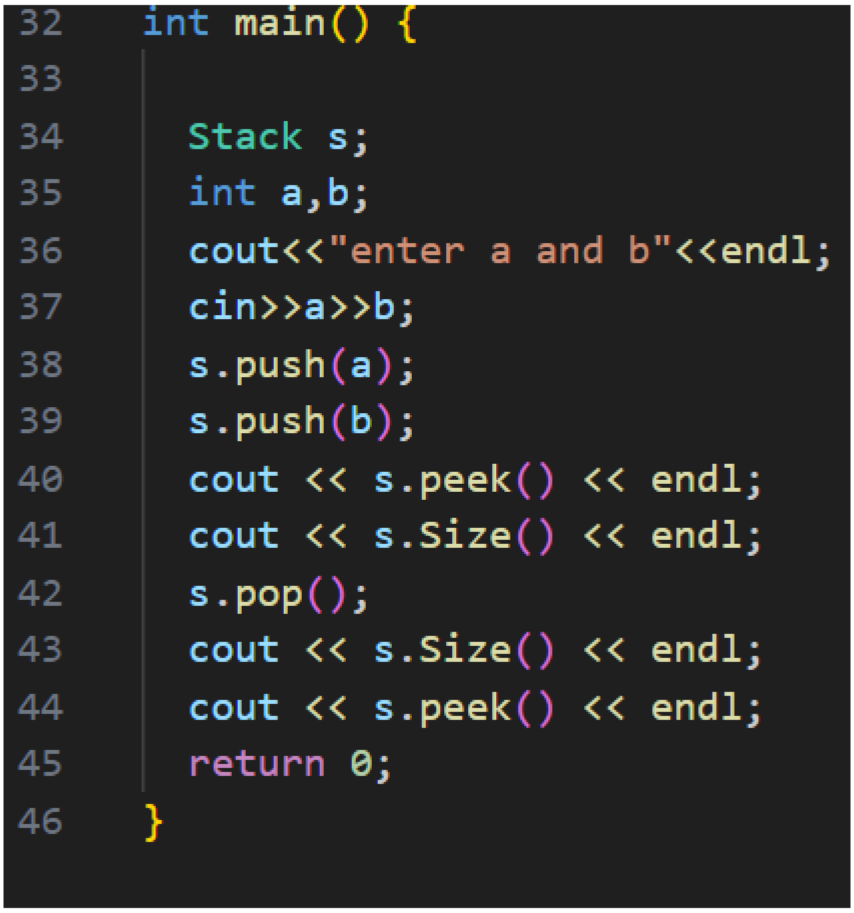
```
enter no of array elements5
1
2
3
4
5
enter k3
 0 for end/n1 for insertathead
 2 for insertattail
 3 for insertatk
 4 for deletehead
 5 for deletetail
6 fordeletekenter choice1
enter value200
200->1->2->3->4->5->HEAD
```

**Q2: Write a C/C++ code to implement stack with following operations using array.**

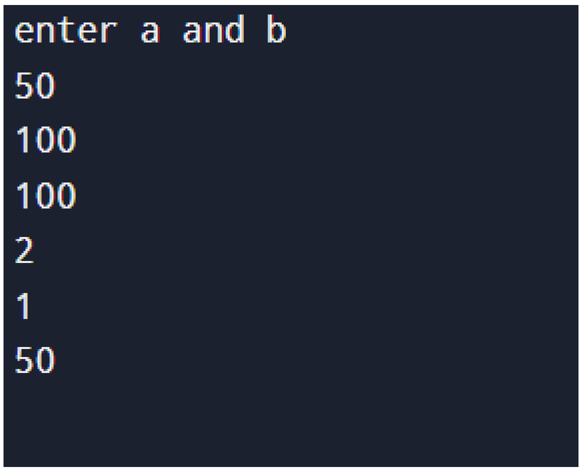
a) create () = Create a stack.
b) push() = Pushing (storing) an element on the stack
c) pop() = Removing (accessing) an element from the stack.
d) peek() = Get the top data element of the stack, without removing it
e) isFull() = Check if stack is full.
f) isEmpty() = Check whether the stack is empty, and return true or false.



```cpp
#include<bits/stdc++.h>
using namespace std;
class Stack {
  int size;
  int * arr;
  int top;
  public:
    Stack() {
      top = -1;
      size = 1000;
      arr = new int[size];
    }
  void push(int x) {
    top++;
    arr[top] = x;
  }
  int pop() {
    int x = arr[top];
    top--;
  }
  int Size() {
    return top + 1;
  }
  int peek() {
    return arr[top];
  }
  bool IsEmpty() {
    return top == -1;
  }

};
```



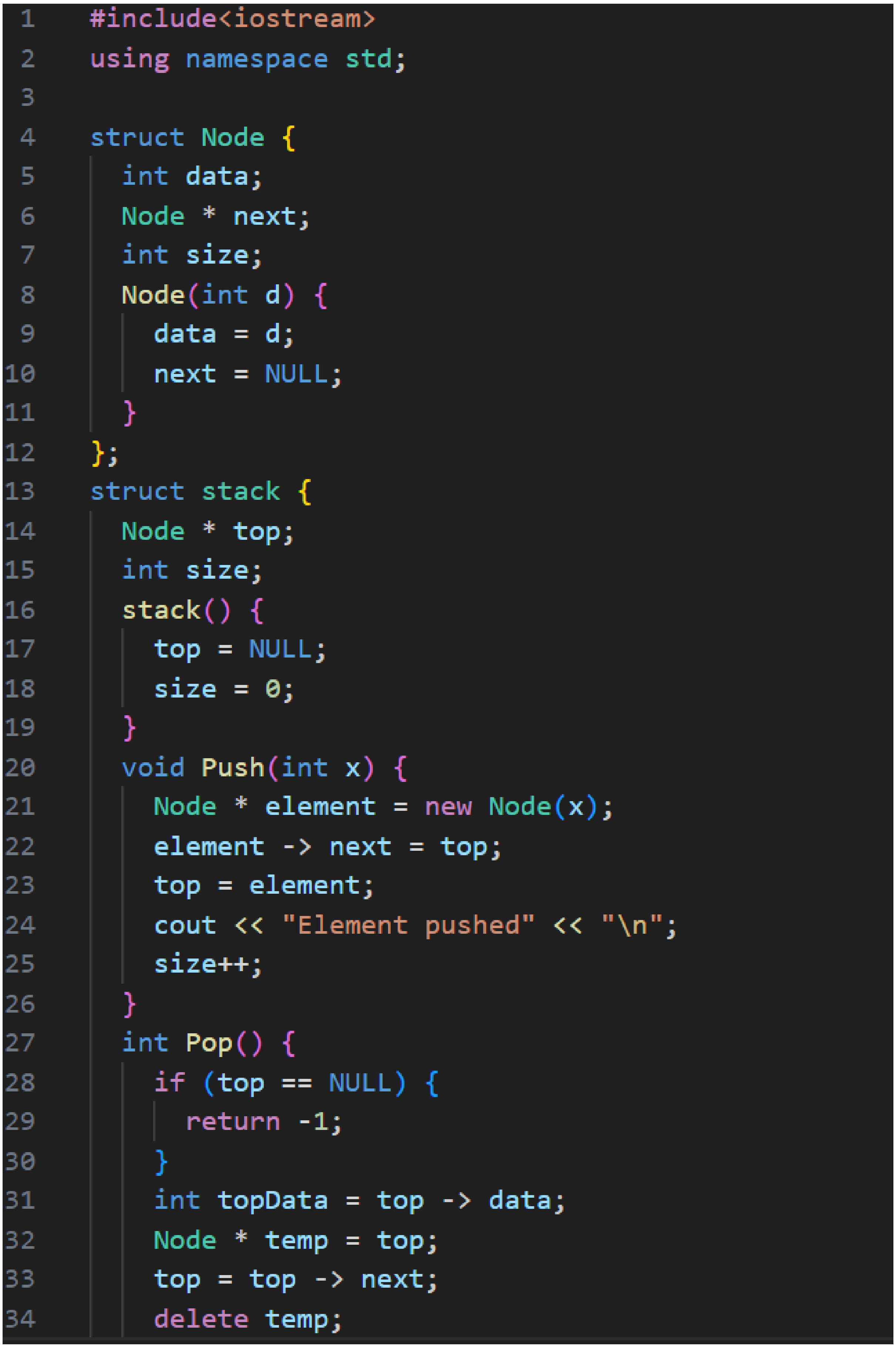
```cpp
int main() {

    Stack s;
    int a,b;
    cout<<"enter a and b"<<endl;
    cin>>a>>b;
    s.push(a);
    s.push(b);
    cout << s.peek() << endl;
    cout << s.Size() << endl;
    s.pop();
    cout << s.Size() << endl;
    cout << s.peek() << endl;
    return 0;
}
```

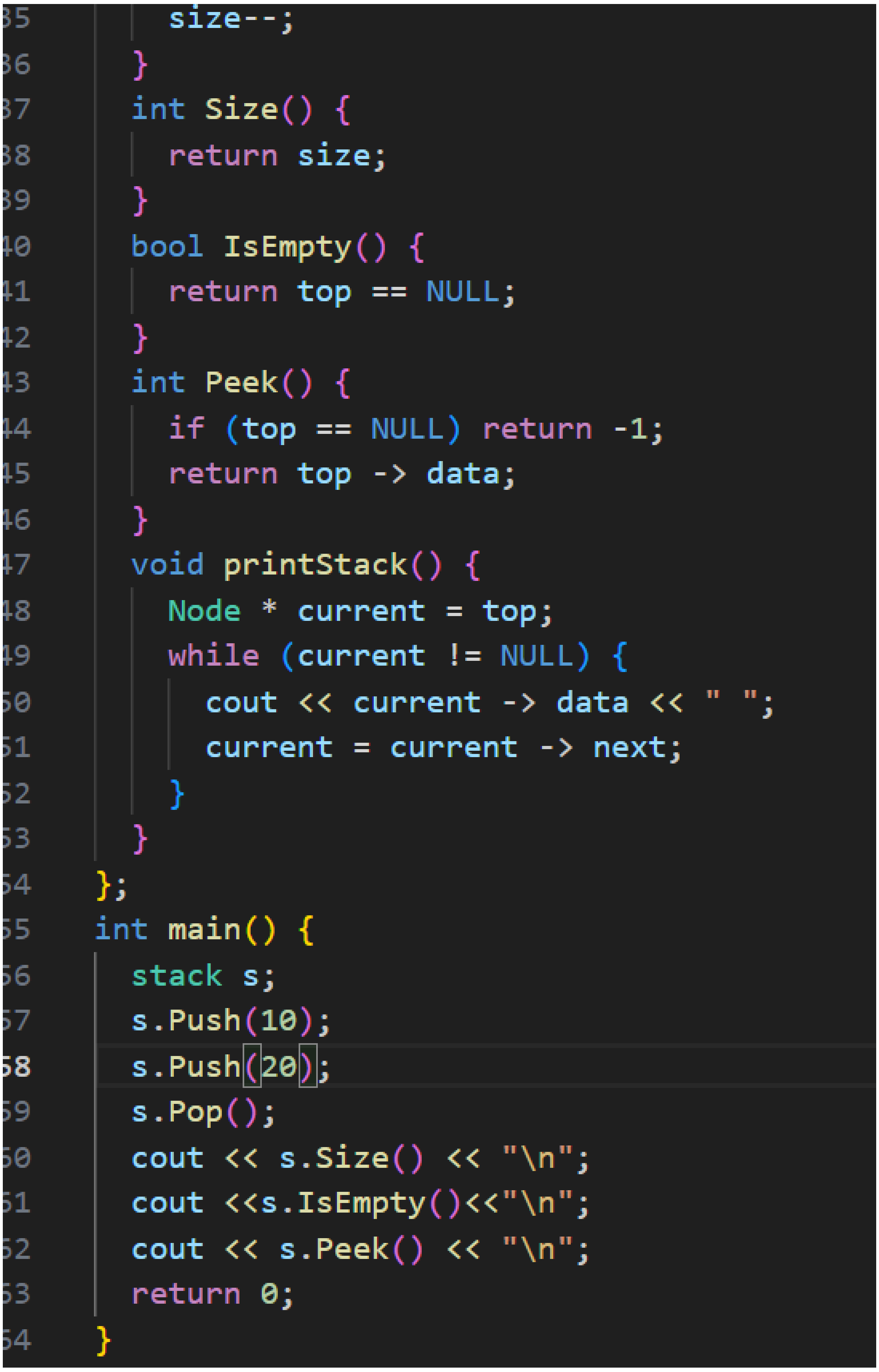


```
enter a and b
50
100
100
2
1
50
```

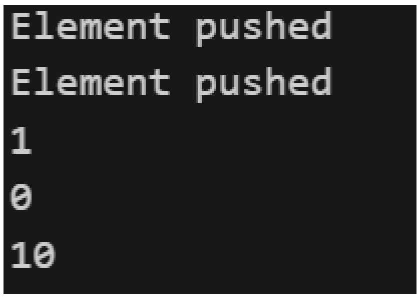
Q3: Write a C/C++ code to implement stack with all the operations defined in

Q2 using Linked list.


```cpp
#include<iostream>
using namespace std;

struct Node {
  int data;
  Node * next;
  int size;
  Node(int d) {
    data = d;
    next = NULL;
  }
};
struct stack {
  Node * top;
  int size;
  stack() {
    top = NULL;
    size = 0;
  }
  void Push(int x) {
    Node * element = new Node(x);
    element -> next = top;
    top = element;
    cout << "Element pushed" << "\n";
    size++;
  }
  int Pop() {
    if (top == NULL) {
      return -1;
    }
    int topData = top -> data;
    Node * temp = top;
    top = top -> next;
    delete temp;
```



```
      size--;
    }
    int Size() {
      return size;
    }
    bool IsEmpty() {
      return top == NULL;
    }
    int Peek() {
      if (top == NULL) return -1;
      return top -> data;
    }
    void printStack() {
      Node * current = top;
      while (current != NULL) {
        cout << current -> data << " ";
        current = current -> next;
      }
    }
};
int main() {
    stack s;
    s.Push(10);
    s.Push(20);
    s.Pop();
    cout << s.Size() << "\n";
    cout <<s.IsEmpty()<<"\n";
    cout << s.Peek() << "\n";
    return 0;
}
```



```
Element pushed
Element pushed
1
0
10
```