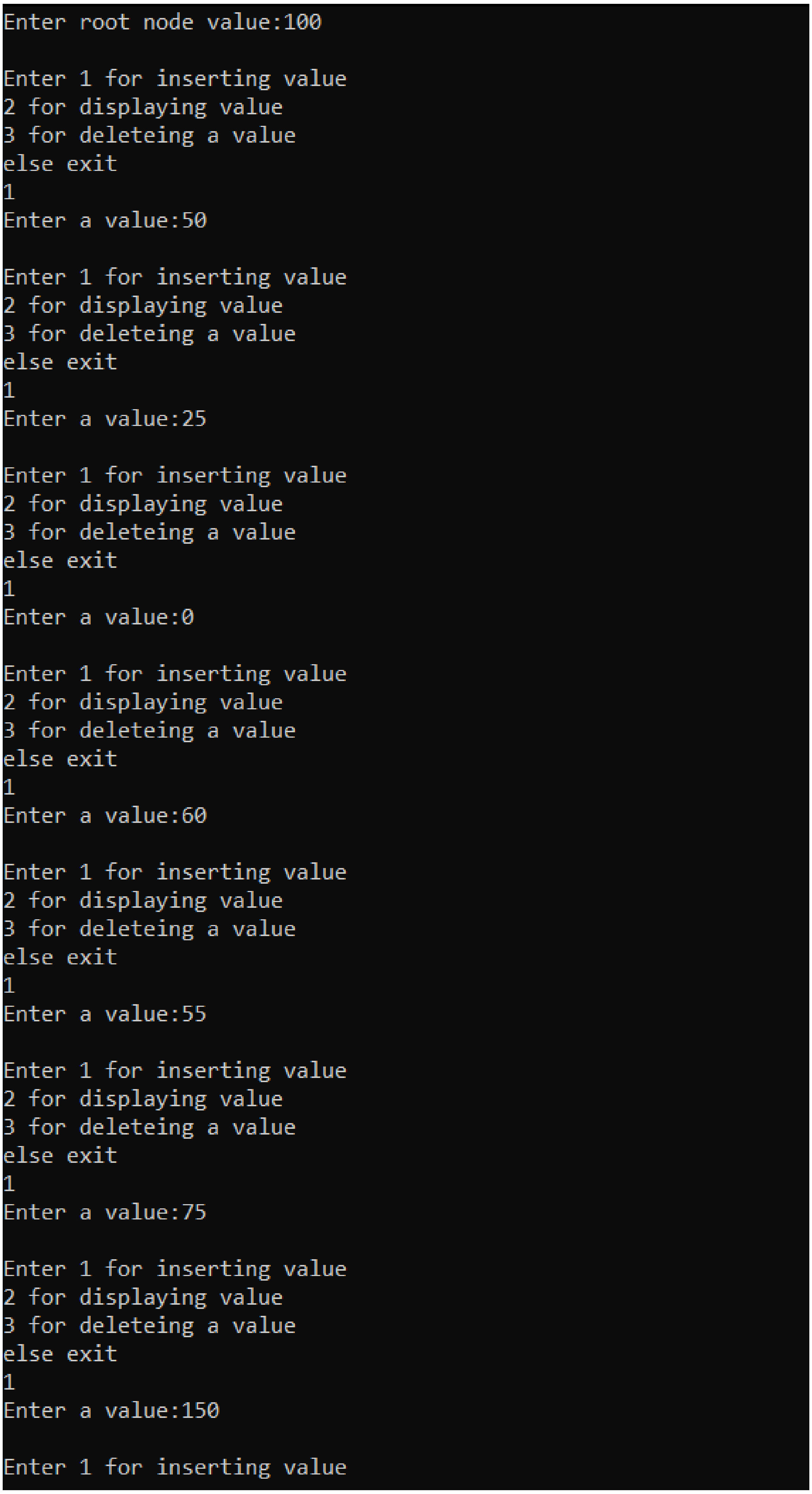
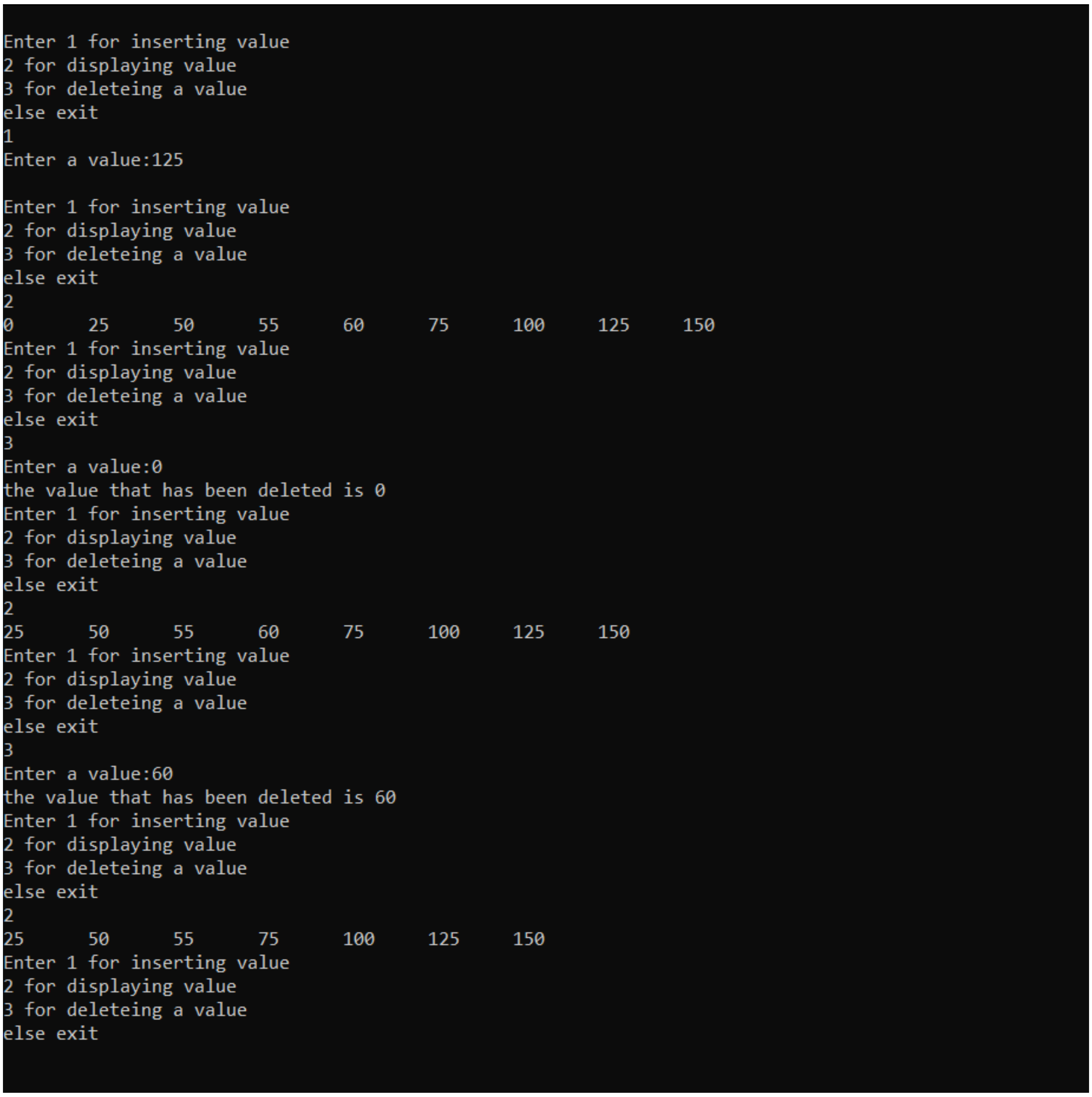
Question 1:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node{
  int val;
  struct node *l;
  struct node *r;
}n;
void init(n *p)
{
  p=NULL;
}
void insert(n *p,int val)
{
  if(p==NULL)
  {
    n *temp=(n*)malloc(sizeof(n));
    temp->val=val;
    temp->l=NULL;
    temp->r=NULL;
    p=temp;
  }
  else if(p->val>=val)
  {
    if(p->l==NULL)
    {
    n *temp=(n*)malloc(sizeof(n));
    temp->val=val;
    temp->l=NULL;
    temp->r=NULL;
    p->l=temp;
    }
    else
    insert(p->l,val);
  }
  else
  {
    if(p->r==NULL)
    {
      n *temp=(n*)malloc(sizeof(n));
      temp->val=val;
      temp->l=NULL;
      temp->r=NULL;
      p->r=temp;
    }
    else
    insert(p->r,val);
  }
}
void inorder(n *p)
{
  if(p==NULL)
  return;
  inorder(p->l);
  printf("%d\t",p->val);
  inorder(p->r);
}
n *f_gc(n *p)
{
  if(p->l==NULL && p->r==NULL)
  return p;
  else
  f_gc(p->l);
}
int delete(n *p,n *p1,int value)
{
  if(p==NULL)
  return -1;
  else if(p->val>value)
  {
    return delete(p->l,p,value);
  }
  else if(p->val<value)
  {
    return delete(p->r,p,value);
  }
  else
  {
    if(p->l==NULL && p->r==NULL)
    {
      if(p1->l==p)
        p1->l=NULL;
      else
        p1->r=NULL;
    }
    else if(p->l==NULL)
    {
      if(p1->l=p)
        p1->l=p->r;
      else
        p1->r=p->r;
      }
      else if(p->r==NULL)
      {
        if(p1->l=p)
          p1->l=p->l;
        else
          p1->r=p->l;
        }
      else
      {
        n *temp=f_gc(p->r);
        temp->l=p->l;
        if(p1->l==p)
          p1->l=p->r;
        if(p1->r==p)
          p1->r=p->r;
      }
      int v=p->val;
      free(p);
      return v;
  }
}
int main()
{
          n *head;
          printf("Enter root node value:");
          n *temp=(n*)malloc(sizeof(n));
          scanf("%d",&(temp->val));
          temp->l=NULL;
          temp->r=NULL;
          head=temp;
          int choice,ch=1,val=0;
          while (ch)
          {
                printf("\nEnter 1 for inserting value\n2 for displaying value\n3 for deleteing a value\nelse
exit\n");
                scanf("%d",&choice);
                switch(choice)
                {
                   case 1:
                        printf("Enter a value:");
                        scanf("%d",&val);
                        insert(head,val);
                        break;
                   case 2:
                        inorder(head);
                        break;
                   case 3:
                        printf("Enter a value:");
                        scanf("%d",&val);
                        if(head->val==val)
                        {
                          n *temp=f_gc(head->r);
                          temp->l=head->l;
                          n *temp1=head;
                          head=head->r;
                          free(temp);
                        }
                        else
                        {
                                printf("the value that has been deleted is %d",delete(head,NULL,val));
                        }
                        break;
                   default:
```

```
                exit(0);
            }
        }
}
```



```
Enter root node value:100

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:50

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:25

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:0

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:60

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:55

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:75

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:150

Enter 1 for inserting value
```



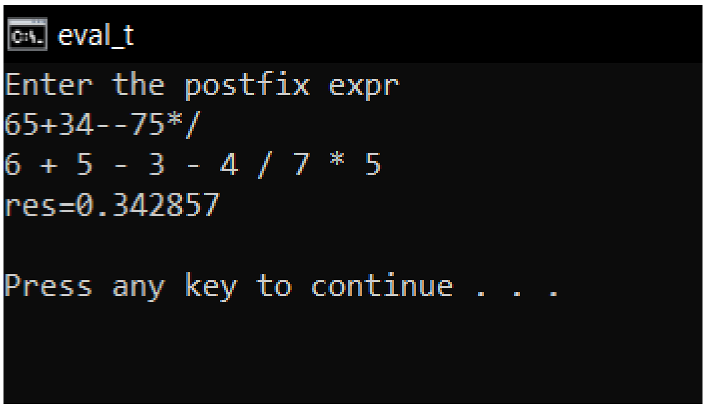
```
Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
1
Enter a value:125

Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
2
0       25      50      55      60      75      100     125     150
Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
3
Enter a value:0
the value that has been deleted is 0
Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
2
25      50      55      60      75      100     125     150
Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
3
Enter a value:60
the value that has been deleted is 60
Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
2
25      50      55      75      100     125     150
Enter 1 for inserting value
2 for displaying value
3 for deleteing a value
else exit
```

Question 2:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 100
typedef struct node_t
{
	char ch;
	struct nr *l;
	struct nr *r;
}nr;
typedef struct tree_t
{
	nr *hea;
}tr;
```

```
typedef struct stack_t
{
        nr *s[MAX];
        int top;
}st;
void init(tr *t)
{
        t->hea=NULL;
}
void init_t(st *p)
{
        p->top=-1;
}
int opr(char ch)
{
        if(ch=='+' || ch=='-' || ch=='*' || ch=='/')
        return 1;
        return 0;
}
void push(st *p,nr *ele)
{
        if(p->top==MAX-1)
        printf("Stack Overflow\n");
        else
        {
                p->top++;
                p->s[p->top]=ele;
        }
}
nr* pop(st *p)
{
        if(p->top==-1)
        return 0;
        else
        {
        nr *t=p->s[p->top];
        p->top--;
        return t;
        }
}
float eval(nr *p)
{
        float res;
        switch(p->ch)
        {
                case '+':
                res=eval(p->l)+eval(p->r);
                break;
                case '-':
                res=eval(p->l)-eval(p->r);
                break;
                case '*':
```

```
		res=eval(p->l)*eval(p->r);
		break;
		case '/':
		res=eval(p->l)/eval(p->r);
		break;
		default:
		return p->ch-'0';
	}
	return res;
}
float evall(tr *t)
{
	return eval(t->hea);
}
int main()
{
	nr *temp;
	tr t;
	init(&t);
	st s;
	init_t(&s);
	char postfix[MAX];
	printf("Enter the postfix expression\n");
	scanf("%s",postfix);
	int i=0;
	while(postfix[i]!='\0')
	{
		temp=(nr*)malloc(sizeof(nr));
		temp->ch=postfix[i];
		temp->l=temp->r=NULL;
		if(!opr(postfix[i]))
		{
		push(&s,temp);
		}
		else
		{
			temp->r=pop(&s);
			temp->l=pop(&s);
			push(&s,temp);
		}
		++i;
	}
	t.hea=pop(&s);
	printf("\nres=%f\n",eval(&t));
	return 0;
}
```



```
Enter the postfix expr
65+34--75*/
6 + 5 - 3 - 4 / 7 * 5
res=0.342857

Press any key to continue . . .
```