Question 1:

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *left, *right;
};

struct node* create(int x) {
    struct node* n = malloc(sizeof(struct node));
    n->data = x;
    n->left = n->right = NULL;
    return n;
}

void preorder(struct node* r) {
    if(r==NULL) return;
    printf("%d ", r->data);
    preorder(r->left);
    preorder(r->right);
}

void inorder(struct node* r) {
    if(r==NULL) return;
    inorder(r->left);
    printf("%d ", r->data);
    inorder(r->right);
}

void postorder(struct node* r) {
    if(r==NULL) return;
    postorder(r->left);
    postorder(r->right);
    printf("%d ", r->data);
}

int main() {
    struct node* root = create(1);
    root->left = create(2);
    root->right = create(3);
    root->left->left = create(4);
    root->left->right = create(5);
    preorder(root);
    printf("\n");
    inorder(root);
    printf("\n");
    postorder(root);
    return 0;
}
```

QUESTION 2:

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *left, *right;
};

struct node* insert(struct node* r, int x) {
    if(r==NULL) {
        r = malloc(sizeof(struct node));
        r->data=x;
        r->left=r->right=NULL;
        return r;
    }
    if(x < r->data) r->left = insert(r->left,x);
    else if(x > r->data) r->right = insert(r->right,x);
    return r;
}

struct node* searchR(struct node* r, int x) {
    if(r==NULL || r->data==x) return r;
    if(x < r->data) return searchR(r->left,x);
    else return searchR(r->right,x);
}

struct node* searchNR(struct node* r, int x) {
    while(r!=NULL && r->data!=x) {
        if(x < r->data) r=r->left;
        else r=r->right;
    }
    return r;
}

struct node* minimum(struct node* r) {
    while(r->left!=NULL) r=r->left;
    return r;
}

struct node* maximum(struct node* r) {
    while(r->right!=NULL) r=r->right;
    return r;
}

struct node* successor(struct node* r, int x) {
    struct node* succ = NULL;
    while(r!=NULL) {
        if(x < r->data) {
            succ = r;
            r = r->left;
        } else r = r->right;
    }
    return succ;
}

struct node* predecessor(struct node* r, int x) {
    struct node* pred = NULL;
    while(r!=NULL) {
        if(x > r->data) {
            pred = r;
            r = r->right;
        } else r = r->left;
    }
    return pred;
}

int main() {
    struct node* root = NULL;
    root = insert(root,50);
    insert(root,30);
    insert(root,70);
    insert(root,20);
    insert(root,40);

    struct node* s = searchNR(root,40);
    if(s) printf("Found\n");

    printf("%d\n", minimum(root)->data);
    printf("%d\n", maximum(root)->data);
    struct node* x = successor(root,30);
    if(x) printf("%d\n", x->data);

    struct node* y = predecessor(root,40);
    if(y) printf("%d\n", y->data);

    return 0;
}
```

QUESTION 3:

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *left, *right;
};

struct node* insert(struct node* r, int x) {
    if(r==NULL) {
        r = malloc(sizeof(struct node));
        r->data=x;
        r->left=r->right=NULL;
        return r;
    }
    if(x < r->data) r->left = insert(r->left,x);
    else if(x > r->data) r->right = insert(r->right,x);
    return r;
}

struct node* minNode(struct node* r) {
    while(r->left!=NULL) r=r->left;
    return r;
}

struct node* delete(struct node* r, int x) {
    if(r==NULL) return r;
    if(x < r->data) r->left = delete(r->left,x);
    else if(x > r->data) r->right = delete(r->right,x);
    else {
        if(r->left==NULL) {
            struct node* t=r->right;
            free(r);
            return t;
        } else if(r->right==NULL) {
            struct node* t=r->left;
            free(r);
            return t;
        }
        struct node* t = minNode(r->right);
        r->data = t->data;
        r->right = delete(r->right,t->data);
    }
    return r;
}

int maxDepth(struct node* r) {
    if(r==NULL) return 0;
    int L = maxDepth(r->left);
    int R = maxDepth(r->right);
    return (L>R?L:R)+1;
}

int minDepth(struct node* r) {
    if(r==NULL) return 0;
    int L = minDepth(r->left);
    int R = minDepth(r->right);
    if(L==0 || R==0) return L+R+1;
    return (L<R?L:R)+1;
}

int main() {
    struct node* root=NULL;
    root=insert(root,50);
    insert(root,40);
    insert(root,60);
    insert(root,30);
    insert(root,70);

    root = delete(root,40);

    printf("%d\n", maxDepth(root));
    printf("%d\n", minDepth(root));
    return 0;
}
```

QUESTION 4 :

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *left, *right;
};

struct node* create(int x) {
    struct node* n = malloc(sizeof(struct node));
    n->data=x;
    n->left=n->right=NULL;
    return n;
}

int isBST(struct node* r, int min, int max) {
    if(r==NULL) return 1;
    if(r->data <= min || r->data >= max) return 0;
    return isBST(r->left,min,r->data) && isBST(r->right,r->data,max);
}

int main() {
    struct node* root = create(5);
    root->left = create(3);
    root->right = create(7);

    if(isBST(root,-99999,99999)) printf("BST\n");
    else printf("Not BST\n");
    return 0;
}
```

QUESTION 5:

```
#include <stdio.h>

void heapify(int a[], int n, int i) {
    int l = 2*i+1, r = 2*i+2, big=i, t;
    if(l<n && a[l]>a[big]) big=l;
    if(r<n && a[r]>a[big]) big=r;
    if(big!=i) {
        t=a[i]; a[i]=a[big]; a[big]=t;
        heapify(a,n,big);
    }
}
```

```
void heapsort(int a[], int n) {
    int i,t;
    for(i=n/2-1;i>=0;i--) heapify(a,n,i);
    for(i=n-1;i>=0;i--) {
        t=a[0]; a[0]=a[i]; a[i]=t;
        heapify(a,i,0);
    }
}

int main() {
    int a[100], n;
    scanf("%d",&n);
    for(int i=0;i<n;i++) scanf("%d",&a[i]);
    heapsort(a,n);
    for(int i=0;i<n;i++) printf("%d ",a[i]);
    return 0;
}
```

QUESTION 6:

```
#include <stdio.h>

int heap[100], n=0;

void insertPQ(int x) {
    heap[n]=x;
    int i=n, p;
    n++;
    while(i>0) {
        p=(i-1)/2;
        if(heap[p] < heap[i]) {
            int t=heap[p]; heap[p]=heap[i]; heap[i]=t;
            i=p;
        } else break;
    }
}

int deletePQ() {
    int x = heap[0];
    heap[0] = heap[n-1];
    n--;
    int i=0, l, r, big, t;
    while(1) {
        l=2*i+1; r=2*i+2; big=i;
        if(l<n && heap[l]>heap[big]) big=l;
        if(r<n && heap[r]>heap[big]) big=r;
        if(big!=i) {
            t=heap[i]; heap[i]=heap[big]; heap[big]=t;
            i=big;
        } else break;
    }
    return x;
}

int main() {
    insertPQ(50);
    insertPQ(20);
    insertPQ(80);
    insertPQ(10);

    printf("%d\n", deletePQ());
    printf("%d\n", deletePQ());
    return 0;
}
```