Q1. Pattern matching

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

int main()
{
    char txt[100];
    char pat[100];
    printf("Give input of a text string : ");
    gets(txt);
    printf("Give input of a pattern string : ");
    gets(pat);
    int M = strlen(pat);
    int N = strlen(txt);


    for (int i = 0; i <= N - M; i++)
    {
        int j;

        for (j = 0; j < M; j++)
            if (txt[i + j] != pat[j])
                break;
        if (j == M)
            printf("Pattern found at index %d \n", i);
    }
    return 0;
}
```

Q2. Insert node at first , middle and last.

```
#include<iostream>
#include<cstdio>
#include<cstdlib>

using namespace std;

struct node{
        int value;
    node *next;
};
struct node *head;

void insertFirst( int a){
   struct node *newNode;
   newNode=(struct node *)malloc(sizeof(struct node));
   newNode->value = a;

   newNode->next = head;
   head = newNode;
}
void insertMiddle(int a, int num){
   struct node *newNode;
   newNode=(struct node *)malloc(sizeof(struct node));
   newNode->value = a;
   newNode->next = NULL;
   struct node *prev = head;
   while (prev->value != num){
       prev = prev->next;
   }
   newNode->next = prev->next;
   prev->next = newNode;
}
```

```
void insertLast(int a){
   struct node *newNode;
   newNode=(struct node *)malloc(sizeof(struct node));
   newNode->value = a;

   struct node *prev = head;
   while (prev->next != NULL)
       prev = prev->next;
   prev->next = newNode;
   newNode->next = NULL;
}
void printList()
{
  if (head == NULL)
         return;
  struct node *cur =  head;
  while (cur != NULL)
  {
       printf("%d \t", cur->value);
       cur = cur->next;
  }
  printf("\n");
}
int main()
{
    int val;
    for(int i=1;i<=5;i++)
    {
        cout<<"Enter a value you want to insert: ";
        cin>>val;
        insertFirst(val);
        printList();
    }
```

```
        insertLast(12);
        printList();
        insertMiddle(19,13);
        printList();
    return 0;
}
```

Q3. Delete node from first, middle and last.

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
using namespace std;

struct node
{
    int value;
    node *next;
};
struct node *head = NULL, *temp = NULL, *tail = NULL, *t1 = NULL, *t2 = NULL, *t3 = NULL;

void delfirst()
{
    head = head -> next;
}
void delmiddle(int n)
{
    int mid = 0;
    t1 = head;
    if(n%2 == 0)
        mid = (n/2)-1;
    else
        mid = n / 2;
    for(int i = 1; i <= mid - 1; i++)
```

```
        t1 = t1 -> next;
    t2 = t1 -> next;
    t3 = t2 -> next;

    t1 -> next = t3;
    delete(t2);
}
void dellast()
{
     temp = head;
    while(temp -> next != tail)
        temp = temp -> next;
    temp -> next = NULL;
    delete(tail);
}
void printList()
{

    temp = head;
    while(temp != NULL)
    {
        cout << temp -> value << " ";
        temp = temp -> next;
    }
    cout << endl;
}
int main()
{
    int val;
    int x;
    cout << "Size of the linked list : ";
    cin >> x;
    for(int i=1; i<=x; i++)
    {
        cout<<"Enter a value you want to insert: ";
```

```
        cin>>val;
        temp = new node();
        temp -> value = val;
        temp -> next = NULL;
        if(head == NULL)
        {
            head = temp;
            tail = temp;
        }
        else
        {
            tail -> next = temp;
            tail = temp;
        }
    }
    printList();
    delfirst();
    x-=1;
    printList();
    dellast();
    x-=1;
    printList();
    delmiddle(x);
    x-=1;
    printList();
    return 0;
}
```

Q4. BST traversal.

```
#include <stdio.h>
#include <stdlib.h>
struct node {
    int data;
    struct node* left;
    struct node* right;
};
void inorder(struct node* root){
    if(root == NULL) return;
    inorder(root->left);
    printf("%d ->", root->data);
    inorder(root->right);
}
void preorder(struct node* root){
    if(root == NULL) return;
    printf("%d ->", root->data);
    preorder(root->left);
    preorder(root->right);
}
void postorder(struct node* root) {
    if(root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ->", root->data);
}
struct node* createNode(int value){
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
struct node* insertLeft(struct node *root, int value) {
```

```
    root->left = createNode(value);
    return root->left;
}
struct node* insertRight(struct node *root, int value){
    root->right = createNode(value);
    return root->right;
}
int main(){
    struct node* root = createNode(1);
    insertLeft(root, 12);
    insertRight(root, 9);

    insertLeft(root->left, 5);
    insertRight(root->left, 6);

    printf("Inorder traversal \n");
    inorder(root);
    printf("\nPreorder traversal \n");
    preorder(root);
    printf("\nPostorder traversal \n");
    postorder(root);
}
```