Data Structures PROGRAMS

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement Bubble sort, to sort a given list of integers in ascending order. |  |
| **Source Code**: (Non-Recursive)  #include<stdio.h>  #include<conio.h>  void main()  {  int i,a[20],temp,n,j;  clrscr();  printf("Enter no of elements:");  scanf("%d",&n);  printf("Enter elements:");  for(i=0;i<n;i++)  scanf("%d",&a[i]);  for(i=0;i<n;i++)  {  for(j=0;j<n-i-1;j++)  {  if(a[j]>a[j+1])  {  temp=a[j];  a[j]=a[j+1];  a[j+1]=temp;  }  }  }  printf("\n Elements after sorting:");  for(i=0;i<n;i++)  printf("%5d",a[i]);  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement Insertion sort, to sort a given list of integers in ascending order. | **Expt No**.: 4(b) |
| **Source Code**: (Non-Recursive)  #include<stdio.h>  #include<conio.h>  void insertion\_sort(int a[],int n)  {  int temp,pos,i;  for(i=0;i<n;i++)  {  temp=a[i];  pos=i;  while(pos>0&&a[pos-1]>temp)  {  a[pos]=a[pos-1];  --pos;  }  a[pos]=temp;  }  printf("\nElements after sorting....\n");  for(i=0;i<n;i++)  printf("%5d",a[i]);  }  void main()  {  int i,n,a[20];  clrscr();  printf("\nEnter the n:");  scanf("%d",&n);  printf("\nEnter the elements:");  for(i=0;i<n;i++)  scanf("%d",&a[i]);  printf("\nElements before sorting:\n");  for(i=0;i<n;i++)  printf("%5d",a[i]);  insertion\_sort(a,n);  getch();  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement quick sort, to sort a given list of integers in ascending order. | **Expt No**.: 4(c) |
|  | |
| **Source Code**: (Recursive)  #include<stdio.h>  #include<conio.h>  void quick\_sort(int,int);  int partition(int,int);  void interchange(int,int);  int a[25];  void main()  {  int i,n;  clrscr();  printf("Enter no of elements:");  scanf("%d",&n);  printf("Enter elements:");  for(i=0;i<n;i++)  scanf("%d",&a[i]);  quick\_sort(0,n);  printf("Elements after sorting:");  for(i=0;i<n;i++)  printf("%3d",a[i]);  getch();  }  int partition(int low,int high)  {  int pivot=a[low];  int up=low,down=high;  do  {  do  {  up=up+1;  }while(a[up]<pivot);  do  {  down=down-1;  }while(a[down]>pivot);  if(up<down)  interchange(up,down);  }while(up<down);  a[low]=a[down];  a[down]=pivot;  return down;  }  void quick\_sort(int low,int high)  {  int pivotpos;  if(low<high)  {  pivotpos=partition(low,high);  quick\_sort(low,pivotpos-1);  quick\_sort(pivotpos+1,high);  }  }  void interchange(int i,int j)  {  int temp;  temp=a[i];  a[i]=a[j];  a[j]=temp;  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement heap sort, to sort a given list of integers in ascending order. | **Expt No**.: 5(a) |
| **Source Code**: (Non-Recursive)  # include <stdio.h>  # include <conio.h>  void main()  {  int b[10],n,i,j,c,p,temp;  clrscr();  printf("\n Enter array size:");  scanf("%d",&n);  printf("\n Enter elements:");  for(i=0;i<n;i++)  scanf("%d",&b[i]);  for(i=1;i<n;i++)  {  c=i;  do  {  p=(c-1)/2;  if(b[p]<b[c])  {  temp=b[p];  b[p]=b[c];  b[c]=temp;  }  else  break;  c=p;  }while(c!=0);  }  for(j=n-1;j>=0;j--)  {  temp=b[0];  b[0]=b[j];  b[j]=temp;  p=0;  do  {  c=2\*p+1;  if((b[c]<b[c+1])&&c<j-1)  c++;  if(b[p]<b[c]&&c<j)  {  temp=b[p];  b[p]=b[c];  b[c]=temp;  }  else  break;  p=c;  }while(c<j);  }  printf("\n Elements after sorting are:");  for(i=0;i<n;i++)  printf("%5d",b[i]);  getch();  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement radix sort, to sort a given list of integers in ascending order. | **Expt No**.: 5(b) |
|  | |
| **Source Code**: (Non-Recursive)  #include<conio.h>  #include<stdio.h>  void radix\_sort(int a[20],int n)  {  int bucket[10][10],ctr[10]={0};  int big,nd,i,j,k,l,div=1,b\_no;  big=a[0];  i=1;  while(i<n)  {  if(a[i]>big)  big=a[i];  i++;  } /\*checking the biggest number\*/  nd=0;  while(big>0)  {  nd=nd+1;  big=big/10;  } /\*checking number of digits in the biggest number\*/  for(i=0;i<n;i++)  {  for(j=0;j<n;j++)  {  b\_no=((a[j]/div)%10);  bucket[b\_no][ctr[b\_no]]=a[j];  ctr[b\_no]=ctr[b\_no]+1;  } /\*Inserting the numbers in to the buckets\*/  div=div\*10;  j=0;  for(k=0;k<10;k++)  {  for(l=0;l<ctr[k];l++)  {  a[j]=bucket[k][l];  j++;  }  ctr[k]=0;  } /\*collecting the elements from the buckets\*/  }  }  void main()  {  int i,n,a[20];  printf("\nEnter n:");  scanf("%d",&n);  printf("\nEnter the elements:");  for(i=0;i<n;i++)  scanf("%d",&a[i]);  printf("\nElements before sorting:\n");  for(i=0;i<n;i++)  printf("%5d",a[i]);  radix\_sort(a,n);  printf("\nElements after sorting:\n");  for(i=0;i<n;i++)  printf("%5d",a[i]);  getch();  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement merge sort, to sort a given list of integers in ascending order. | **Expt No**.: 5(c) |
| **Source Code**: (Recursive)  #include<stdio.h>  #include<conio.h>  void merge(int a[],int start,int mid,int end)  {  int b[20],k=0,i,j;  i=start;  j=mid+1;  while((i<mid+1)&&(j<end+1))  {  if(a[i]<a[j])  {  b[k]=a[i];  i++;  }  else  {  b[k]=a[j];  j++;  }  k++;  }  while(i<mid+1)  {  b[k]=a[i];  k++;  i++;  }  while(j<end+1)  {  b[k]=a[j];  j++;  k++;  }  for(i=0;i<k;i++)  {  a[start]=b[i];  start++;  }  }  void merge\_sort(int a[20],int start,int end)  {  int mid;  if(start<end)  {  mid=(start+end)/2;  merge\_sort(a,start,mid);  merge\_sort(a,mid+1,end);  merge(a,start,mid,end);  }  }  void main()  {  int i,n,a[20],start=0,end;  clrscr();  printf("\nEnter the n:");  scanf("%d",&n);  printf("\nEnter the elements:");  for(i=0;i<n;i++)  scanf("%d",&a[i]);  printf("\nElements before sort:\n");  for(i=0;i<n;i++)  printf("%5d",a[i]);  end=n-1;  merge\_sort(a,start,end);  printf("\nElements after sort:\n");  for(i=0;i<n;i++)  printf("%5d",a[i]);  getch();  } | |
| **Aim**:Write C programs that implement stack (its operations) using arrays. | **Expt No**.: 6(a) |
| **Source Code**:  # include <stdio.h>  # include <conio.h>  # define MAX 10  int stack[MAX];  int top=0;  int menu()  {  int ch;  printf("\n \t\t\t\tStack using arrays:");  printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  printf("\n 1.Push");  printf("\n 2.Pop");  printf("\n 3.Display");  printf("\n 4.Quit");  printf("\n Enter your choice:");  scanf("%d",&ch);  return ch;  }  void display()  {  int i;  if(top==0)  {  printf("\n Stack is empty");  return;  }  else  {  printf("The elements in the stack are:");  for(i=0;i<top;i++)  printf("%c",stack[i]);  }  }  void pop()  {  if(top==0)  {  printf("\n Stack underflow");  return;  }  else  printf("\n The popped element from stack is: %c",stack[--top]);  }  void push()  {  int c;  if(top==MAX)  {  printf("\n Stack overflow");  return;  }  else  {  printf("\n Enter charecter:");  scanf("%c",&c);  stack[top]=c;  top=top+1;  printf("\n charecter inserted successfully");  }  }  void main()  {  int ch;  do  {  ch=menu();  switch(ch)  {  case 1:push();  break;  case 2:pop();  break;  case 3:display();  break;  default:printf("\n UVE BEEN QUITTED OUT");  break;  }  }while(ch==1||ch==2||ch==3);  getch();  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement stack (its operations) using linked list. | **Expt No**.: 6(b) |
| **Source Code**:  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  struct stack  {  int data;  struct stack \*next;  };  typedef struct stack node;  node \*start = NULL;  node \*top = NULL;  node\* getnode()  {  node \*temp;  temp=(node \*)malloc(sizeof(node));  printf("\n enter data : ");  scanf("%d",&temp -> data);  temp -> next=NULL;  return temp;  }  void push(node \*newnode)  {  node \*temp;  if(newnode==NULL)  {  Printf ("\n stack overflow... ");  return;  }  if(start == NULL)  {  start = newnode;  top = newnode;  }  else  {  temp = start;  while(temp -> next!= NULL)  temp = temp -> next;  temp -> next =newnode;  top = newnode;  }  Printf ("\n\n\t data pushed into stack...");  }  void pop()  {  node \*temp;  if(top == NULL)  {  printf("\n\n\t stack overflow...");  return;  }  temp = start;  if(start -> next == NULL)  {  printf("\n\n\t popped element is %d",top -> data);  start = NULL;  free(top);  top = NULL;  }  else  {  while(temp -> next!=top)  {  temp = temp -> next;  }  temp -> next = NULL;  printf("\n\n\t popped element is %d", top -> data);  free(top);  top = temp;  }  }  void display()  {  node \*temp;  if(top == NULL)  {  printf("\n\n\t stack is empty...");  }  else  {  temp = start;  Printf ("\n\n\t elements in the stack..\n");  printf("%5d",temp -> data);  while(temp!= top)  {  temp = temp -> next;  printf("%5d",temp -> data);  }  }  }  int menu()  {  int ch;  printf("\n\t STACK OPERATIONS USING POINTERS : ");  printf("\n ----\*\*\*\*\*\*----- ");  printf("\n 1. Push ");  Printf ("\n 2. Pop ");  printf("\n 3. Display ");  printf("\n 4. Quit");  printf("\n enter your choice : ");  scanf("%d",&ch);  return ch;  }  void main()  {  int ch;  node \*newnode;  do  {  ch=menu();  switch(ch)  {  case 1 :  newnode=getnode();  push(newnode);  break;  case 2 :  pop();  break;  case 3 :  display();  break;  case 4 :  return;  }  getch();  }while(1);  } | |

|  |  |
| --- | --- |
| **Aim**:Write a C program that uses Stack operations to Convert infix expression into postfix expression. | **Expt No**.: 7(a) |
| **Source Code**:  #include<stdio.h>  #include<string.h>  char opstack[50];  char infix[50];  char postfix[50];  int i,j,top=0;  void pop();  void push(char);  int lesspriority(char,char);  void main()  {  char ch;  printf("enter infix expression:\n");  gets(infix);  while((ch=infix[i++])!='\0')  {  switch(ch)  {  case ' ':break;  case '(':  case '+':  case '-':  case '\*':  case '/':  case '^':  case '%':  push(ch);  break;  case ')':  pop();  break;  default:  postfix[j]=ch;  j++;  }  }  while(top>=0)  {  postfix[j]=opstack[--top];  j++;  }  postfix[--j]='\0';  printf("\n infix expression:%s",infix);  printf("\n postfix expression:%s",postfix);  }  int lesspriority(char op,char op\_at\_stack)  {  int k;  int pv1;  int pv2;  char operators[]={'+','-','\*','/','%','^','('};  int priority\_value[]={0,0,1,1,2,3,4};  if(op\_at\_stack=='(')  return 0;  for(k=0;k<6;k++)  {  if(op==operators[k])  pv1=priority\_value[k];  }  for(k=0;k<6;k++)  {  if(op\_at\_stack==operators[k])  pv2=priority\_value[k];  }  if(pv1<pv2)  return 1;  else  return 0;  }  void push(char op)  {  if(top==0)  {  opstack[top]=op;  top++;  }  else  {  if(op!='(')  {  while(lesspriority(op,opstack[top-1])==1 && top>0)  {  postfix[j]=opstack[--top];  j++;  }  }  opstack[top]=op;  top++;  }  }  void pop()  {  While (opstack [--top]! ='(')  {  postfix[j]=opstack[top];  j++;  }  } | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement Queue (its operations) using arrays. | **Expt No**.: 7(b) |
|  | |
| **Source Code**:  # include <stdio.h>  # include <conio.h>  # define MAX 4  int queue[MAX];  int front=0,rear=0;  int menu()  {  int ch;  printf("\n \t\t\t\t\tQueue using arrays:");  printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  printf("\n 1.Insertion");  printf("\n 2.Deletion");  printf("\n 3.Display");  printf("\n 4.QUIT");  printf("\n Enter your choice:");  scanf("%d",&ch);  return ch;  }  void insertq()  {  int data;  if(rear==MAX)  {  printf("\n Queue is full");  return;  }  else  {  printf("\n Enter data:");  scanf("%d",&data);  queue[rear]=data;  rear=rear+1;  printf("\n Data entered successfully");  }  }  void deleteq()  {  if(front==rear)  {  printf("\n Queue is empty");  return;  }  else  {  printf("\n The deleted element from the queue is:%d",queue[front]);  front=front+1;  }  }  void displayq()  {  int i;  if(front==rear)  {  printf("\n No elements in the queue");  return;  }  else  {  printf("\n The elements in the queue are:");  for(i=front;i<rear;i++)  printf("%5d",queue[i]);  }  }  void main()  {  int ch;  do  {  ch=menu();  switch(ch)  {  case 1:insertq();  break;  case 2:deleteq();  break;  case 3:displayq();  break;  default:printf("\n U HAVE QUITED OUT");  }  }while(ch==1||ch==2||ch==3);  getch();  }  /\* | |

|  |  |
| --- | --- |
| **Aim**:Write C programs that implement Queue (its operations) using linked list. | **Expt No**.: 7(c) |
| **Source Code**:  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  struct queue  {  int data;  struct queue \*next;  };  typedef struct queue node;  node \*front = NULL;  node \*rear = NULL;  node\* getnode()  {  node \*temp;  temp=(node \*)malloc(sizeof(node));  printf("\n enter data : ");  scanf("%d",&temp -> data);  temp -> next=NULL;  return temp;  }  void insertQ()  {  node \*newnode;  newnode = getnode();  if(newnode==NULL)  {  printf("\n Queue is full... ");  return;  }  if(front == NULL)  {  front = newnode;  rear = newnode;  }  else  {  rear -> next = newnode;  rear = newnode;  }  printf("\n\n\t data inserted into Queue...");  }  void deleteQ()  {  node \*temp;  if(front == NULL)  {  printf("\n\n\t Empty Queue...");  return;  }  temp = front;  front = front -> next;  printf("\n\n\t popped element is %d",temp -> data);  free(temp);  }  void displayQ()  {  node \*temp;  if(front == NULL)  {  printf("\n\n\t Empty Queue...");  }  else  {  temp = front;  printf("\n\n\t elements in the Queue...\n");  while(temp!= NULL)  {  printf("%5d",temp -> data);  temp = temp -> next;  }  }  }  int menu()  {  int ch;  printf("\n\t QUEUE OPERATIONS USING POINTERS : ");  printf("\n ----\*\*\*\*\*\*----- ");  printf("\n 1. Insert ");  printf("\n 2. Delete ");  printf("\n 3. Display ");  printf("\n 4. Quit");  printf("\n enter your choice : ");  scanf("%d",&ch);  return ch;  }  void main()  {  int ch;  node \*newnode;  do  {  ch=menu();  switch(ch)  {  case 1 :  insertQ();  break;  case 2 :  deleteQ();  break;  case 3 :  displayQ();  break;  case 4 :  return;  }  getch();  }while(1);  } | |

|  |  |
| --- | --- |
| **Aim**:Write a C program that uses functions to create a singly linked list and perform insertion, deletion and traversing operations on a singly linked list. | **Expt No**.: 8 |
| **Source Code**:  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  struct linklist  {  int data;  struct linklist \*next;  };  typedef struct linklist node;  node \*start=NULL;  int menu()  {  int ch;  printf("\n\t \*\*\*\*\*IMPLIMENTATION OF SINGLE LINKED LIST\*\*\*\*\*");  printf("\n\t ---------------------------------------\n");  printf("\n\t 1.Create list");  printf("\n\t 2.Traverse the list(left to right)");  printf("\n\t 3.Traverse the list(right to left)");  printf("\n\t 4.Number of nodes");  printf("\n\t 5.Insertion at Begining");  printf("\n\t 6.Insertion at End");  printf("\n\t 7.Insertion at Middle");  printf("\n\t 8.Deletion at Beginning");  printf("\n\t 9.Deletion at End");  printf("\n\t 10.Deletion at Middle");  printf("\nEnter your choice:");  scanf("%d",&ch);  return ch;  }  node\* getnode()  {  node \*newnode;  newnode=(node\*)malloc(sizeof(node));  printf("Enter data:\n");  scanf("%d",&newnode->data);  newnode->next=NULL;  return newnode;  }  int countnode(node\*start)  {  if(start==NULL)  return 0;  else  return 1+countnode(start->next);  }  void createlist(int n)  {  int i;  node \*newnode;  node \*temp;  for(i=0;i<n;i++)  {  newnode=getnode();  if(start==NULL)  {  start=newnode;  }  else  {  temp=start;  while(temp->next!=NULL)  temp=temp->next;  temp->next=newnode;  }  }  }  void traverse()  {  node \*temp;  temp=start;  printf("The contents of the list(left to right)\n");  if(start==NULL)  {  printf("\n Empty list");  return;  }  else  {  while(temp!=NULL)  {  printf("%d\t",temp->data);  temp=temp->next;  }  }  printf("X");  }  void rev\_traverse(node \*start)  {  if(start==NULL)  {  return;  }  else  {  rev\_traverse(start->next);  printf("%d\t",start->data);  }  }  void insert\_at\_beg()  {  node \*newnode;  newnode=getnode();  if(start==NULL)  {  start=newnode;  }  newnode->next=start;  start=newnode;  }  void insert\_at\_end()  {  node \*newnode,\*temp;  newnode=getnode();  if(start==NULL)  {  start=newnode;  }  else  {  temp=start;  while(temp->next!=NULL)  temp=temp->next;  temp->next=newnode;  }  }  void insert\_at\_mid()  {  node \*newnode,\*pre,\*temp;  int pos,ctr=1,nodectr;  printf("Enter position:");  scanf("%d",&pos);  nodectr=countnode(start);  if(pos>1 && pos<nodectr)  {  newnode=getnode();  temp=pre=start;  while(ctr<pos)  {  pre=temp;  temp=temp->next;  ctr++;  }  pre->next=newnode;  newnode->next=temp;  }  else  {  printf("\nNot a middle position");  }  }  void del\_at\_beg()  {  node \*temp;  if(start==NULL)  {  printf("List is empty");  return;  }  else  {  temp=start;  start=temp->next;  free(temp);  printf("Node is deleted");  }  }  void del\_at\_end()  {  node \*pre,\*temp;  if(start==NULL)  {  printf("List is empty");  return;  }  else  {  temp=start;  while(temp->next!=NULL)  {  pre=temp;  temp=temp->next;  }  pre->next=NULL;  free(temp);  printf("\Node deleted");  }  }  void del\_at\_mid()  {  int pos,ctr=1,nodectr;  node \*temp,\*pre;  nodectr=countnode(start);  if(start==NULL)  {  printf("List is empty");  return;  }  else  {  printf("Enter position:");  scanf("%d",&pos);  if(pos>1 && pos<nodectr)  {  pre=temp=start;  while(ctr<pos)  {  pre=temp;  temp=temp->next;  ctr++;  }  pre->next=temp->next;  free(temp);  }  else  printf("Not a mid position");  }  }  void main(void)  {  int ch,n;  clrscr();  while(1)  {  ch=menu();  switch(ch)  {  case 1:  if(start==NULL)  {  printf("Enter the number of nodes you want to create:");  scanf("%d",&n);  createlist(n);  printf("List is created");  break;  }  else  {  printf("List is already created:");  break;  }  case 2:traverse();  break;  case 3:  printf("The contents of the list(left to right):\n");  rev\_traverse(start);  printf("X");  break;  case 4: printf("Number of nodes:%d",countnode(start));  break;  case 5: insert\_at\_beg();  break;  case 6: insert\_at\_end();  break;  case 7: insert\_at\_mid();  break;  case 8: del\_at\_beg();  break;  case 9: del\_at\_end();  break;  case 10: del\_at\_mid();  break;  case 11: exit(0);  }  }  } | |
| **Input & Output**:  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:1  Enter the number of nodes you want to create:5  Enter data:  1  Enter data:  2  Enter data:  3  Enter data:  4  Enter data:  5  List is created  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  1 2 3 4 5 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:3  The contents of the list(left to right):  5 4 3 2 1 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:4  Number of nodes:5  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:5  Enter data:  0  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  0 1 2 3 4 5 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:6  Enter data:  6  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  0 1 2 3 4 5 6 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:7  Enter position:4  Enter data:  7  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  0 1 2 7 3 4 5 6 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:8  Node is deleted  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  1 2 7 3 4 5 6 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:9  Node deleted  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  1 2 7 3 4 5 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:10  Enter position:3  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:2  The contents of the list(left to right)  1 2 3 4 5 X  \*\*\*\*\*IMPLIMENTATION OF LINKED LIST\*\*\*\*\*  ---------------------------------------  1.Create list  2.Traverse the list(left to right)  3.Traverse the list(right to left)  4.Number of nodes  5.Insertion at Begining  6.Insertion at End  7.Insertion at Middle  8.Deletion at Beginning  9.Deletion at End  10.Deletion at Middle  Enter your choice:11 | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to add two large integers which are represented in linked list fashion. | **Expt No**.: 9(a) |
|  | |
| **Source Code**:  /\*Aim:To perform addition of two large integers using linked list\*/  #include<stdio.h>  #include<conio.h>  #include<stdlib.h>  struct number  {  int data;  struct number \*next,\*prev;  };  typedef struct number node;  node \*ptr,\*temp,\*t,\*st1=NULL,\*st2=NULL,\*st3=NULL,\*p1,\*p2,\*t1=NULL,\*t2=NULL,\*t3=NULL;  node\* getnode()  {  node\*newnode;  newnode=(node\*)malloc(sizeof(node));  newnode->next=NULL;  newnode->prev=NULL;  return newnode;  }  void main()  {  char num1[50],num2[50],ch;  int i;  printf("\nEnter two numbers:\n ");  gets(num1);  gets(num2);  for(i=0;num1[i]!='\0';i++)  {  ch=num1[i];  ptr=getnode();  ptr->data=ch-48;  if(st1==NULL)  {  st1=ptr;  t1=ptr;  }  else  {  t1->next=ptr;  ptr->prev=t1;  t1=ptr;  }  }  for(i=0;num2[i]!='\0';i++)  {  ch=num2[i];  ptr=getnode();  ptr->data=ch-48;  if(st2==NULL)  {  st2=ptr;  t2=ptr;  }  else  {  t2->next=ptr;  ptr->prev=t2;  t2=ptr;  }  }  p1=t1;p2=t2;  while(p1!=NULL && p2!=NULL)  {  ptr=getnode();  ptr->data=p1->data+p2->data;  if(ptr->data>9)  {  if(p1->prev!=NULL)  {  p1->prev->data=p1->prev->data+1;  t=p1->prev;  while(t->data>9&&t!=NULL)  {  t->prev->data=t->prev->data+1;  t->data=(t->data)%10;  t=t->prev;  }  }  else if(p2->prev!=NULL)  {  p2->prev->data=p2->prev->data+1;  }  else  {  ptr->prev=getnode();  ptr->prev->data=1;  ptr->prev->next=ptr;  ptr->next=st3;  st3->prev=ptr->prev;  st3=ptr->prev;  }  ptr->data=(ptr->data)%10;  }  if(ptr->prev==NULL)  {  if(t3==NULL)  {  t3=ptr;  st3=ptr;  }  else  {  ptr->next=st3;  st3->prev=ptr;  st3=ptr;  }  }  p1=p1->prev;  p2=p2->prev;  }  while(p1!=NULL)  {  ptr=getnode();  ptr->data=p1->data;  ptr->next=st3;  st3->prev=ptr;  st3=ptr;  p1=p1->prev;  }  while(p2!=NULL)  {  ptr=getnode();  ptr->data=p2->data;  ptr->next=st3;  st3->prev=ptr;  st3=ptr;  p2=p2->prev;  }  printf("\nThe sum of two numbers is:\n");  ptr=st3;  while(ptr!=NULL)  {  printf("%d",ptr->data);  ptr=ptr->next;  }  } | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to reverse elements of a single linked list. | **Expt No**.: 9(b) |
| **Source Code**:  /\*aim: to reverse elements of a single linked list\*/  #include<stdio.h>  #include<stdlib.h>  struct node  {  int data;  struct node \*link;  };  typedef struct node node;  node \*start=NULL;  node \* getnode()  {  node \* ptr;  ptr=(node \*)malloc(sizeof(node));  printf("Enter data:\n");  scanf("%d",&ptr->data);  ptr->link=NULL;  return ptr;  }  void create\_ll(int N)  {  node \*ptr,\*t;  int i;  printf("Enter the number of nodes you want:\n");  scanf("%d",&N);  for(i=0;i<N;i++)  {  ptr=getnode();  if(start==NULL)  {  start=ptr;  }  else  {  t=start;  while(t->link!=NULL)  {  t=t->link;  }  t->link=ptr;  }  }  }  void reverse\_ll(node \*start)  {  if(start==NULL)  {  return;  }  else  {  reverse\_ll(start->link);  printf("%d ",start->data);  }  }  int menu()  {  int ch;  printf("1.Create list\n");  printf("2.Reverse the list\n");  printf("3.Exit\n");  printf("Enter your choice:");  scanf("%d",&ch);  return ch;  }  void main()  {  int n,ch;  while(1)  {  ch=menu();  switch(ch)  {  case 1:  create\_ll(n);  printf("List created\n");  break;  case 2:  printf("The list in reverse order:");  reverse\_ll(start);  printf("\n");  break;  case 3:  exit(0);  }  }  } | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to store a polynomial expression in memory using linked list. | **Expt No**.: 9(c) |
| **Source Code**:  #include<stdio.h>  #include<stdlib.h>  #include<conio.h>  #include<malloc.h>  struct link  {  float coef;  int expo;  struct link \*next;  };  typedef struct link node;  node \*getnode()  {  node \*temp;  temp=(node\*)malloc(sizeof(node));  printf("Enter coefficient:");  fflush(stdin);  scanf("%f",&temp->coef);  printf("Enter exponent:");  fflush(stdin);  scanf("%d",&temp->expo);  temp->next=NULL;  return temp;  }  node \*create\_poly(node \*p)  {  char ch;  node \*temp,\*newnode;  while(1)  {  printf("Do you want polynomial node(y/n):\n");  ch=getch();  if(ch=='n')  break;  newnode=getnode();  if(p==NULL)  p=newnode;  else  {  temp=p;  while(temp->next!=NULL)  temp=temp->next;  temp->next=newnode;  }  }  return p;  }  void display(node \*p)  {  node \*t=p;  while(t!=NULL)  {  printf("+%.f",t->coef);  printf("x^%d",t->expo);  t=t->next;  }  }  void main()  {  node \*poly1=NULL;  Printf ("Enter First Polynomial..(in ascending order of exponent)\n");  poly1=create\_poly(poly1);  printf("Polynomial 1:");  display(poly1);  } | |
| **Input & Output**:  Enter First Polynomial..(in ascending order of exponent)  Do you want polynomial node(y/n):Y  Enter coefficient:12  Enter exponent:3  Do you want polynomial node(y/n):Y  Enter coefficient:45  Enter exponent:4  Do you want polynomial node(y/n):N  Polynomial 1:+12x^3+45x^4 | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to representation the given Sparse matrix using arrays. | **Expt No**.: 9(d) |
| **Source Code**:  #include<stdio.h>  void main()  {  int matrix[20][20],m,n,total\_elements,total\_zeros=0,i,j;  printf("Enter number of Rows and Columns:");  scanf("%d %d",&m,&n);  total\_elements=m\*n;  printf("Enter data for Sparse matrix\n");  for(i=0;i<m;i++)  {  for(j=0;j<n;j++)  {  scanf("%d",&matrix[i][j]);  if(matrix[i][j]==0)  {  total\_zeros++;  }  }  }  if(total\_zeros>total\_elements/2)  {  printf("Given Matrix is a Sparse matrix\n");  printf("The Representation of Sparse matrix\n");  printf("Row\tColumn\tValue\n");  for(i=0;i<m;i++)  {  for(j=0;j<n;j++)  {  if(matrix[i][j]!=0)  {  printf(" %d \t %d \t %d\n",i,j,matrix[i][j]);  }  }  }  }  else  Printf ("Given matrix is not a Sparse matrix...”);  } | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to representation the given Sparse matrix using linked list. | **Expt No**.: 9(e) |
| **Source Code**:  #include<stdio.h>  #include<stdlib.h>  typedef struct sparse\_matrix  {  int row,col,data;  struct sparse\_matrix \*link;  }node;  node\*start=NULL;  void attach\_at\_end(node\*);  node\*createnode(int,int,int);  void traverse();  void main()  {  int i,j,m,n,a[20][20],t\_zeroes=0,t\_elements;  node \*ptr;  printf("Enter the number of rows & columns :");  scanf("%d%d",&m,&n);  printf("\nEnter the data for sparse matrix:");  for(i=0;i<m;i++)  {  for(j=0;j<n;j++)  {  scanf("%d",&a[i][j]);  if(a[i][j]==0)  t\_zeroes++;  }  }  t\_elements=m\*n;  if(t\_elements/2<t\_zeroes)  {  printf("\nThe given matrix is sparse matrix");  printf("\nThe representation of the matrix is:\n");  for(i=0;i<m;i++)  {  for(j=0;j<n;j++)  {  if(a[i][j]!=0)  {  ptr=createnode(i,j,a[i][j]);  attach\_at\_end(ptr);  }  }  }  printf("\nrow\tcolumn\tvalue");  traverse();  }  else  printf("It is not a sparse matrix");  }  node\*createnode(int r,int c,int d)  {  node \*ptr;  ptr=(node\*)malloc(sizeof(node));  ptr->row=r;  ptr->col=c;  ptr->data=d;  ptr->link=NULL;  return ptr;  }  void attach\_at\_end(node \*ptr)  {  node \*t;  t=start;  if(start==NULL)  start=ptr;  else  {  while(t->link!=NULL)  t=t->link;  t->link=ptr;  }  }  void traverse()  {  node \*t;  t=start;  while(t!=NULL)  {  printf("\n%d\t%d\t%d",t->row,t->col,t->data);  t=t->link;  }  } | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to create a Binary Tree of integers, traverse the tree in preorder, inorder and postorder and also print the number of leaf nodes and height of the tree. | **Expt No**.:10(a & b) |
| **Source Code**:  #include<stdio.h>  #include<stdlib.h>  struct tree  {  struct tree \*lchild;  char data[10];  struct tree \*rchild;  };  typedef struct tree node;  node \*tree[50];  int ctr;  node \* getnode()  {  node \*temp;  temp=(node\*)malloc(sizeof(node));  printf("Enter data:");  scanf("%s",temp->data);  temp->lchild=NULL;  temp->rchild=NULL;  return temp;  }  void create\_binarytree()  {  int j,i=0;  printf("\nHow many nodes you want:");  scanf("%d",&ctr);  tree[0]=getnode();  j=ctr;  j--;  do  {  if(j>0)  {  tree[i\*2+1]=getnode();  tree[i]->lchild=tree[i\*2+1];  j--;  }  if(j>0)  {  tree[i\*2+2]=getnode();  j--;  tree[i]->rchild=tree[i\*2+2];  }  i++;  }while(j>0);  }  void inorder(node \* root)  {  if(root!=NULL)  {  inorder(root->lchild);  printf("%3s",root->data);  inorder(root->rchild);  }  }  void preorder(node \* root)  {  if(root!=NULL)  {  printf("%3s",root->data);  preorder(root->lchild);  preorder(root->rchild);  }  }  void postorder(node \* root)  {  if(root!=NULL)  {  postorder(root->lchild);  postorder(root->rchild);  printf("%3s",root->data);  }  }  void levelorder()  {  int j;  for(j=0;j<ctr;j++)  {  if(tree[j]!=NULL)  printf("%3s",tree[j]->data);  }  }  void print\_leaf(node \*root)  {  if(root!=NULL)  {  if(root->lchild==NULL && root->rchild==NULL)  printf("%3s",root->data);  print\_leaf(root->lchild);  print\_leaf(root->rchild);  }  }  int height(node \* root)  {  if(root==NULL)  {  return 0;  }  if(root->lchild==NULL && root->rchild==NULL)  return 0;  else  return (1 + max(height(root->lchild),height(root->rchild)));  }  void main()  {  create\_binarytree();  printf("\nInorder traversal");  inorder(tree[0]);  printf("\nPreorder traversal");  preorder(tree[0]);  printf("\nPostorder traversal");  postorder(tree[0]);  printf("\nLevel order traversal");  levelorder();  printf("\nLeaf nodes:");  print\_leaf(tree[0]);  printf("\nHeight of the Tree :%d",height(tree[0]));  } | |
| **Input & Output**:  How many nodes you want:5  Enter data:1  Enter data:3  Enter data:2  Enter data:4  Enter data:5  inorder traversal: 4 3 5 1 2  preorder traversal: 1 3 4 5 2  postorder traversal: 4 5 3 2 1  levelorder traversal: 1 3 2 4 5  leaf nodes:4 5 3  height of tree:2 | |

|  |  |
| --- | --- |
| **Aim**:Write a C non-recursive program to create a Binary Tree of integers, traverse the tree in preorder, inorder and postorder. | **Expt No**.: 10(c) |
| **Source Code**:  /\*aim:To perform non-resursive programs of inorder,preorder,postorder traversals of binary tree\*/  #include<stdio.h>  #include<stdlib.h>  struct tree  {  struct tree \*lchild;  char data[10];  struct tree \*rchild;  };  typedef struct tree node;  node \*a[50],\*b[50];  int node\_ctr,top=-1,top1=-1;  node \*getnode()  {  node \*temp;  temp=(node\*)malloc(sizeof(node));  printf("Enter data:");  scanf("%s",temp->data);  temp->lchild=NULL;  temp->rchild=NULL;  return temp;  }  void create\_binarytree(node \*root)  {  char option;  node\_ctr=1;  if(root!=NULL)  {  printf("Node %s has left subtree(Y/N):",root->data);  fflush(stdin);  scanf("%c",&option);  if(option=='Y'||option=='y')  {  root->lchild=getnode();  node\_ctr++;  create\_binarytree(root->lchild);  }  else  {  root->lchild=NULL;  create\_binarytree(root->lchild);  }  printf("Node %s has right subtree(Y/N):",root->data);  fflush(stdin);  scanf("%c",&option);  if(option=='Y'||option=='y')  {  root->rchild=getnode();  node\_ctr++;  create\_binarytree(root->rchild);  }  else  {  root->rchild=NULL;  create\_binarytree(root->rchild);  }  }  }  void push(node \*y)  {  top++;  a[top]=y;  }  node \*pop()  {  return a[top--];  }  void nrinorder(node \*root)  {  node \*l;  l=root;  do  {  while(l!=NULL)  {  push(l);  l=l->lchild;  }  while(top>-1)  {  l=pop();  printf("%s ",l->data);  if(l->rchild!=NULL)  {  l=l->rchild;  break;  }  else  l=NULL;  }  }while(l!=NULL);  }  void nrpreorder(node \*root)  {  node \*l;  l=root;  do  {  printf("%s ",l->data);  if(l->rchild!=NULL)  {  push(l->rchild);  }  l=l->lchild;  if(l==NULL&&top>-1)  l=pop();  }while(l!=NULL);  }  void nrpostorder(node \*root)  {  node \*l;  l=root;  do  {  while(l!=NULL)  {  push(l);  if(l->rchild!=NULL)  {  push(l->rchild);  b[++top1]=l->rchild;  }  l=l->lchild;  }  do  {  l=pop();  if(l!=b[top1])  {  printf("%s ",l->data);  }  else  {  top1--;  break;  }  }while(top>-1);  }while(l!=NULL&&top>-1);  }  int menu()  {  int ch;  printf("\n1.Create Binary Tree\n");  printf("2.Inorder Traversal\n");  printf("3.Preorder Traversal\n");  printf("4.Postorder Traversal\n");  printf("5.Quit\n");  printf("Enter your choice:");  scanf("%d",&ch);  return ch;  }  void main()  {  int ch;  node \*root=NULL;  do  {  ch=menu();  switch(ch)  {  case 1:  if(root==NULL)  {  root=getnode();  create\_binarytree(root);  }  else  {  printf("Tree already created\n");  }  break;  case 2:  printf("Inorder Traversal:");  nrinorder(root);  break;  case 3:  printf("Preorder Traversal:");  nrpreorder(root);  break;  case 4:  printf("Postorder Traversal:");  nrpostorder(root);  break;  case 5:  exit(0);  }  }  while(1);  } | |
| **Input & Output**:  1.Create Binary Tree  2.Inorder Traversal  3.Preorder Traversal  4.Postorder Traversal  5.Quit  Enter your choice:1  Enter data:A  Node A has left subtree(Y/N):Y  Enter data:B  Node B has left subtree(Y/N):Y  Enter data: D  Node D has left subtree(Y/N):N  Node D has right subtree(Y/N):N  Node B has right subtree(Y/N):Y  Enter data:E  Node E has left subtree(Y/N):N  Node E has right subtree(Y/N):N  Node A has right subtree(Y/N):Y  Enter data:C  Node C has left subtree(Y/N):Y  Enter data:F  Node F has left subtree(Y/N):N  Node F has right subtree(Y/N):N  Node C has right subtree(Y/N):Y  Enter data:G  Node G has left subtree(Y/N):N  Node G has right subtree(Y/N):N  1.Create Binary Tree  2.Inorder Traversal  3.Preorder Traversal  4.Postorder Traversal  5.Quit  Enter your choice:2  Inorder Traversal:D B E A F C G  1.Create Binary Tree  2.Inorder Traversal  3.Preorder Traversal  4.Postorder Traversal  5.Quit  Enter your choice:3  Preorder Traversal:A B D E C F G  1.Create Binary Tree  2.Inorder Traversal  3.Preorder Traversal  4.Postorder Traversal  5.Quit  Enter your choice:4  Postorder Traversal:D E B F G C A  1.Create Binary Tree  2.Inorder Traversal  3.Preorder Traversal  4.Postorder Traversal  5.Quit  Enter your choice:5 | |

|  |  |
| --- | --- |
| **Aim**:Write a C program to check the balance property of a tree. | **Expt No**.: 10(d) |
| **Source Code**:  #include<stdio.h>  #include<stdlib.h>  struct tree  {  struct tree \*lchild;  char data[10];  struct tree \*rchild;  };  typedef struct tree node;  node \*Q[50];  int node\_ctr;  node \*getnode()  {  node \*temp;  temp=(node\*)malloc(sizeof(node));  printf("Enter data:");  scanf("%s",temp->data);  temp->lchild=NULL;  temp->rchild=NULL;  return temp;  }  void create\_binarytree(node \*root)  {  char option;  node\_ctr=1;  if(root!=NULL)  {  printf("Node %s has left subtree(Y/N):",root->data);  fflush(stdin);  scanf("%c",&option);  if(option=='Y'||option=='y')  {  root->lchild=getnode();  node\_ctr++;  create\_binarytree(root->lchild);  }  else  {  root->lchild=NULL;  create\_binarytree(root->lchild);  }  printf("Node %s has right subtree(Y/N):",root->data);  fflush(stdin);  scanf("%c",&option);  if(option=='Y'||option=='y')  {  root->rchild=getnode();  node\_ctr++;  create\_binarytree(root->rchild);  }  else  {  root->rchild=NULL;  create\_binarytree(root->rchild);  }  }  }  int check\_balance(node \*root)  {  if(root==NULL)  return -1;  else  return(1+max(check\_balance(root->lchild),check\_balance(root->rchild)));  }  int menu()  {  int ch;  printf("1.Create Binary Tree\n");  printf("2.Check balance property\n");  printf("3.Quit\n");  printf("Enter your choice:");  scanf("%d",&ch);  return ch;  }  void main()  {  int ch,a,b;  node \*root=NULL;  do  {  ch=menu();  switch(ch)  {  case 1:  if(root==NULL)  {  root=getnode();  create\_binarytree(root);  }  else  {  printf("Tree already created\n");  }  break;  case 2:  a=check\_balance(root->lchild);  b=check\_balance(root->rchild);  if(a==b)  printf("The tree is balanced\n");  else  printf("Tree is not balanced\n");  break;  case 3:  exit(0);  }  }  while(1);  } | |
| **Input & Output**:  1.Create Binary Tree  2.Check balance property  3.Quit  Enter your choice:1  Enter data:1  Node 1 has left subtree(Y/N):y  Enter data:2  Node 2 has left subtree(Y/N):y  Enter data:3  Node 3 has left subtree(Y/N):n  Node 3 has right subtree(Y/N):n  Node 2 has right subtree(Y/N):y  Enter data:4  Node 4 has left subtree(Y/N):n  Node 4 has right subtree(Y/N):n  Node 1 has right subtree(Y/N):y  Enter data:5  Node 5 has left subtree(Y/N):y  Enter data:6  Node 6 has left subtree(Y/N):n  Node 6 has right subtree(Y/N):n  Node 5 has right subtree(Y/N):y  Enter data:7  Node 7 has left subtree(Y/N):n  Node 7 has right subtree(Y/N):n  1.Create Binary Tree  2.Check balance property  3.Quit  Enter your choice:2  The tree is balanced  1.Create Binary Tree  2.Check balance property  3.Quit  Enter your choice:3  1.Create Binary Tree  2.Check balance property  3.Quit  Enter your choice:1  Enter data:1  Node 1 has left subtree(Y/N):y  Enter data:2  Node 2 has left subtree(Y/N):y  Enter data:3  Node 3 has left subtree(Y/N):n  Node 3 has right subtree(Y/N):n  Node 2 has right subtree(Y/N):y  Enter data:4  Node 4 has left subtree(Y/N):n  Node 4 has right subtree(Y/N):n  Node 1 has right subtree(Y/N):y  Enter data:5  Node 5 has left subtree(Y/N):n  Node 5 has right subtree(Y/N):n  1.Create Binary Tree  2.Check balance property  3.Quit  Enter your choice:2  Tree is not balanced  1.Create Binary Tree  2.Check balance property  3.Quit  Enter your choice:3 | |