/\* Practical No.01

Title: Write a program to find the length of string and print its reverse.

Name of Programmer: Vishal Pravin Jatti Roll No: MC23F14F026

\*/

#include<stdio.h> #include<string.h> #include<conio.h> int main()

{

char st[10]; int n,i; char \*q; char \*ps; ps=st; q=ps;

printf("Enter a string:"); while(\*ps=getchar()!='\n')

{

ps++;

}

while(\*ps!='\n')

{

ps++;

}

n=ps-q;

printf("\nLength of string=%d",n);

--ps;

printf("\nString in reverse order is:"); for(i=0;i<n;i++)

{

putchar(\*ps); ps--;

}

getch(); return 0;

}

## OUTPUT:

Enter a string:MCA

Length of string=3

String in reverse order is:ACM

/\* Practical No.02

Title: Write a program for array of structure. Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

#include<stdio.h> #include<conio.h> struct invent

{

char \*name[10]; int number;

int price;

};

int main()

{

struct invent product[3]; struct invent \*p;

int i=1; p=&product;

printf(“Enter the product name,number and price:\n”);

for(p=product;p<product+3;p++)

{

printf(“Product %d\n”,i); scanf(“%s%d%d”,pname,&pnumber,&pprice); i++;

}

for(p=product;p<product+3;p++)

{

printf(“\n%s%d%d” ,pname,&pnumber,&pprice);

}

getch(); return 0;

}

## OUTPUT:

Enter the product name,number and price: Product 1:

Fan 2

1000

Product 2: LED

3

5000

Product 3: PC

6

29999

Fan 2 1000

LED 3 5000

PC 6 29999

/\* Practical No.03

Title: Write a program to implement stack operation on stack.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

#include<stdio.h> #include<conio.h> #define maxsize 10

void push(struct stack \*p,int x); int pop(struct stack \*ps);

struct stack

{

}s;

int top;

int item[maxsize];

void main()

{

int choice,ele,\*i; struct stack \*ps; s.top=-1;

printf(“Available choices are:\nChoice 1:PUSH \nChoice 2:POP

\nChoice 3:Display stack \nChoice 4:Exit \n”); while(choice!=4)

{

printf(“\nEnter your choice:”); scanf(“%d”,&choice); if(choice==1)

{

printf(“\nEnter the element to be inserted:”); scanf(“%d”,&ele);

push(&s,ele);

}

if(choice==2)

{

\*i=pop(&s);

printf(“\nThe deleted item is:%d\n”, \*i);

}

if(choice==3)

{

showstack(&s);

}

if(choice==4)

{

exit(0);

}

}

getch();

}

void push(struct stack \*ps,int x)

{

pstop++; psitem[pstop]=x;

}

int pop(struct stack \*ps)

{

int y; y=psitem[pstop]; pstop--;

return y;

}

void showstack(struct stack \*ps)

{

int i; if(pstop==-1)

{

printf(“Stack is empty…”);

}

else

{

printf(“Elements of stack:-\n”);

for(i=pstop;i>=0;i--)

{

printf(“%d\n”,psitem[i]);

}

}

}

## OUTPUT:

Choice 1:PUSH Choice 2:POP

Choice 3:Display stack Choice 4:Exit

Enter your choice:1 Element to be inserted:4 Enter your choice:1 Element to be inserted:7 Enter your choice:1 Element to be inserted:9 Enter your choice:3 Elements of stack:-

9

7

4

Enter your choice:4

/\* Practical No.04

Title: Write a program for evaluation of postfix expression using stack.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

#include<stdio.h> #include<conio.h> #include<math.h> struct stack

{

int top;

int operand[20];

};

void push(struct stack\*,int); void pop(struct stack \*); void main()

{

struct stack s; char str1[40];

int i=0,result,op1,op2; s.top;

printf(“Enter an expression in postfix form:\n”); while((str1[i]=getchar())!=’\n’)

{

if(isdigit(str1[i]))

{

push(&s,str1[i]-‘0’);

}

else

{

op2=s.operand[s.top]; pop(&s); op1=s.operand[s.top]; pop(&s); switch(str1[i])

{

case’+’:result=op1+op2;

break;

case’-‘:result=op1-op2; break;

case’\*’:result=op1\*op2; break;

case’/’:result=op1/op2;

break;

}

printf(“\nIntermediate result=%d”,result);

push(&s,result);

}

i++;

}

printf(“\nThe value of expression=%d”,result);

getch();

}

void push(struct stack \*ps,int n)

{

pstop++; psoperand[pstop]=n;

}

void pop(struct stack \*ps)

{

pstop--;

}

## OUTPUT:

Enter an expression in postfix form: 35+42-\*

Intermediate result=8 Intermediate result=2 Intermediate result=16

The value of expression =16

/\* Practical No 05

Title: Write a program to convert infix expression to postfix.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

#include<stdio.h> #include<string.h> char stack[50]; int top=-1;

void post(char infix[]); void push(char);

char pop(); int main()

{

char infix[25];

printf("\nENTER THE INFIX EXPRESSION = ");

gets(infix); post(infix);

}

void push(char symb)

{

if(top>=49)

{

}

else

{

}

}

printf("\nSTACK OVERFLOW"); return;

top=top+1; stack[top]=symb;

char pop()

{

char item; if(top==-1)

{

}

else

{

}

printf("\nSTACK IS EMPTY"); return(0);

item=stack[top]; top--;

return(item);

}

int preced(char ch)

{

if(ch==47)

{

return(5);

}

else if(ch==42)

{

return(4);

}

else if(ch==43)

{

return(3);

}

else

return(2);

}

void post(char infix[])

{

int l;

int index=0,pos=0; char symbol,temp; char postfix[40]; l=strlen(infix); push('#'); while(index<l)

{

symbol=infix[index]; switch(symbol)

{

case '(': push(symbol); break;

case ')': temp=pop();

while(temp!='(')

{

}

break;

postfix[pos]=temp; pos++;

temp=pop();

case '+':

case '-':

case '\*':

case '/':

case '^': while(preced(stack[top])>=preced(symbol))

{

temp=pop(); postfix[pos]=temp; pos++;

}

}

index++;

}

push(symbol); break;

default: postfix[pos++]=symbol; break;

while(top>0)

{

temp=pop(); postfix[pos++]=temp;

}

postfix[pos++]='\0'; puts(postfix); return;

}

# OUTPUT:

ENTER THE INFIX EXPRESSION = (a+b)\*c

ab+c\*

Process returned 0 (0x0) execution time : 7.887 s Press any key to continue.

/\* Practical No 06

Title: Write a program for singly and doubly linked list.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

**//6.A: C program to perform singly linked list.**

#include<stdio.h> #include<stdlib.h> struct Node;

typedef struct Node \* PtrToNode; typedef PtrToNode List;

typedef PtrToNode Position; struct Node

{

int e;

Position next;

};

void Insert(int x, List l, Position p)

{

Position TmpCell;

TmpCell = (struct Node\*) malloc(sizeof(struct Node)); if(TmpCell == NULL)

printf("Memory out of space\n"); else

{

TmpCell->e = x;

TmpCell->next = p->next; p->next = TmpCell;

}

}

int isLast(Position p)

{

return (p->next == NULL);

}

Position FindPrevious(int x, List l)

{

Position p = l;

while(p->next != NULL && p->next->e != x) p = p->next;

return p;

}

void Delete(int x, List l)

{

Position p, TmpCell;

p = FindPrevious(x, l); if(!isLast(p))

{

TmpCell = p->next;

p->next = TmpCell->next; free(TmpCell);

}

else

printf("Element does not exist!!!\n");

}

void Display(List l)

{

printf("The list element are :: "); Position p = l->next;

while(p != NULL)

{

printf("%d -> ", p->e);

p = p->next;

}

}

void Merge(List l, List l1)

{

int i, n, x, j;

Position p;

printf("Enter the number of elements to be merged :: "); scanf("%d",&n);

for(i = 1; i <= n; i++)

{

p = l1; scanf("%d", &x);

for(j = 1; j < i; j++) p = p->next;

Insert(x, l1, p);

}

printf("The new List :: "); Display(l1);

printf("The merged List ::"); p = l;

while(p->next != NULL)

{

p = p->next;

}

p->next = l1->next; Display(l);

}

int main()

{

int x, pos, ch, i;

List l, l1;

l = (struct Node \*) malloc(sizeof(struct Node)); l->next = NULL;

List p = l;

printf("LINKED LIST IMPLEMENTATION OF LIST ADT.");

do

{

printf("\n1. INSERT\t 2. DELETE\t 3. MERGE\t 4. PRINT\t 5.

QUIT\nEnter the choice :: "); scanf("%d", &ch); switch(ch)

{

case 1:

p = l;

printf("Enter the element to be inserted :: "); scanf("%d",&x);

printf("Enter the position of the element :: "); scanf("%d",&pos);

for(i = 1; i < pos; i++)

{

p = p->next;

}

Insert(x,l,p); break;

case 2:

p = l;

printf("Enter the element to be deleted :: "); scanf("%d",&x);

Delete(x,p);

Break;

case 3:

l1 = (struct Node \*) malloc(sizeof(struct Node)); l1->next = NULL;

Merge(l, l1); break;

case 4:

Display(l); break;

}

}

while(ch<5); return 0;

}

# OUTPUT:

LINKED LIST IMPLEMENTATION OF LIST ADT.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. INSERT Enter the  Enter the | 2. DELETE choice :: 1  element to be inserted | | | 3.  :: | MERGE  11 | 4. | | PRINT | 5. | QUIT |
| Enter the | position of the element :: 1 | | | | |  | |  |  |  |
| 1. INSERT Enter the Enter the | 2. DELETE choice :: 1  element to be inserted | | | 3.  :: | MERGE 12 | 4. | | PRINT | 5. | QUIT |
| Enter the | position of the element :: 2 | | | | |  | |  |  |  |
| 1. INSERT Enter the Enter the | 2. DELETE choice :: 1  element to be inserted | | | 3.  :: | MERGE 13 | 4. | | PRINT | 5. | QUIT |
| Enter the | position of the element :: 3 | | | | |  | |  |  |  |
| 1. INSERT Enter the Enter the | 2. DELETE choice :: 1  element to be inserted | | | 3.  :: | MERGE 14 | 4. | | PRINT | 5. | QUIT |
| Enter the | position of the element :: 4 | | | | |  | |  |  |  |
| 1. INSERT Enter the | choice | 2. DELETE  :: 4 | | 3. | MERGE | 4. | | PRINT | 5. | QUIT |
| The list element  1. INSERT | | are :: 11  2. DELETE | -> 12 | ->  3. | 13 ->  MERGE | 14 -> | 4. | PRINT | 5. | QUIT |
| Enter the choice | | :: 5 |  |  |  |  |  |  |  |  |

Process returned 0 (0x0) execution time : 27.663 s Press any key to continue.

### //6.2: C program to perform doubly linked list.

#include<stdio.h> #include<stdlib.h> struct Node;

typedef struct Node \* PtrToNode; typedef PtrToNode List;

typedef PtrToNode Position; struct Node

{

int e;

Position previous;

Position next;

};

void Insert(int x, List l, Position p)

{

Position TmpCell;

TmpCell = (struct Node\*) malloc(sizeof(struct Node)); if(TmpCell == NULL)

printf("Memory out of space\n"); else

{

TmpCell->e = x; TmpCell->previous = p;

TmpCell->next = p->next; p->next = TmpCell;

}

}

int isLast(Position p)

{

return (p->next == NULL);

}

Position Find(int x, List l)

{

Position p = l->next;

while(p != NULL && p->e != x) p = p->next;

return p;

}

void Delete(int x, List l)

{

Position p, p1, p2; p = Find(x, l); if(p != NULL)

{

p1 = p -> previous; p2 = p -> next;

p1 -> next = p -> next; if(p2 != NULL)

p2 -> previous = p -> previous;

}

else

printf("Element does not exist!!!\n");

}

void Display(List l)

{

printf("The list element are :: "); Position p = l->next;

while(p != NULL)

{

printf("%d -> ", p->e);

p = p->next;

}

}

void main()

{

int x, pos, ch, i;

List l, l1;

l = (struct Node \*) malloc(sizeof(struct Node)); l->previous = NULL;

l->next = NULL;

List p = l;

printf("DOUBLY LINKED LIST IMPLEMENTATION OF LIST ADT\n");

do

{

printf("\n1. INSERT\t 2. DELETE\t 3. FIND\t 4. PRINT\t 5.

QUIT\nEnter the choice :: "); scanf("%d", &ch); switch(ch)

{

case 1:

p = l;

printf("Enter the element to be inserted :: "); scanf("%d",&x);

printf("Enter the position of the element :: "); scanf("%d",&pos);

for(i = 1; i < pos; i++)

{

p = p->next;

}

Insert(x,l,p); break;

case 2:

p = l;

printf("Enter the element to be deleted :: "); scanf("%d",&x);

Delete(x,p); break;

case 3:

p = l;

printf("Enter the element to be searched :: "); scanf("%d",&x);

p = Find(x,p); if(p == NULL)

printf("Element does not exist!!!\n"); else

printf("Element exist!!!\n"); break;

case 4:

Display(l); break;

}

}

while(ch<5); return 0;

}

# OUTPUT:

DOUBLY LINKED LIST IMPLEMENTATION OF LIST ADT

1. INSERT 2. DELETE 3. FIND 4. PRINT 5. QUIT Enter the choice :: 1

Enter the element to be inserted :: 22 Enter the position of the element :: 1

1. INSERT 2. DELETE 3. FIND 4. PRINT 5. QUIT Enter the choice :: 1

Enter the element to be inserted :: 33 Enter the position of the element :: 2

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. INSERT Enter the Enter the  Enter the | 2. DELETE 3. FIND choice :: 1  element to be inserted :: 44  position of the element :: 3 | | | | 4. | PRINT | 5. | QUIT |
| 1. INSERT | 2. | DELETE | 3. | FIND | 4. | PRINT | 5. | QUIT |
| Enter the | choice | :: 3 |  |  |  |  |  |  |
| Enter the | element to be | | searched | :: 22 |  |  |  |  |

Element exist!!!

1. INSERT 2. DELETE 3. FIND 4. PRINT 5. QUIT Enter the choice :: 4

The list element are :: 22 -> 33 -> 44 ->

1. INSERT 2. DELETE 3. FIND 4. PRINT 5. QUIT Enter the choice :: 5

Process returned 5 (0x5) execution time : 33.785 s Press any key to continue.

/\* Practical No 07

Title:Write a program to perform circular queue using linked list.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

#include<stdio.h> #include<stdlib.h> #define que struct queue #define pf printf #define sf scanf

struct queue

{

int info;

struct queue \*link;

};

que \*front=NULL,\*rear=NULL; int count=0;

void push(int n)

{

que \*newnode;

newnode=(struct queue\*)malloc(sizeof(struct queue)); newnode->info=n;

newnode->link=NULL; if(count==0)

front=newnode; else

rear->link=newnode; rear=newnode;

rear->link=front; count++;

}

int pop(void)

{

int n;

que \*temp; if(count==0)

return (-1);

count--; if(front==rear)

{

n=front->info; free(front); front=NULL; rear=NULL;

}

else

{

temp= front ;

n = temp-> info ;

front = front -> link ; rear -> link = front ; free ( temp ) ;

}

return n;

}

void display(void)

{

que \*temp; int i; if(count==0)

pf("Empty"); else

{

temp=front; for(i=0;i<count;i++)

{

pf("%d ",temp->info); temp=temp->link;

}

}

pf("\n");

}

int size(void)

{

return count;

}

int main()

{

int n,ch=10; while(ch!=0)

{

pf("\nWhat do you want to do??\n"); pf("1.Push\t2.Pop\t3.Size Of Queue\t 4.Display\t0.Exit\n"); sf("%d",&ch);

switch(ch)

{

case 1:

{

pf("What no. do you want to push in queue\n"); sf("%d",&n);

push(n); break;

}

case 2:

{

n=pop();

if(n==-1)

pf("Queue is empty\n"); else

pf("Number poped from queue is %d\n",n); break;

}

case 3:

{

n=size();

pf("Size of queue is %d\n",n); break;

}

case 4:

{

pf("Queue is -->> "); display();

}

case 0:

break; default:

pf("Wrong Choice\n"); break;

}

}

}

# OUTPUT:

What do you want to do??

1.Push 2.Pop 3.Size Of Queue 4.Display 0.Exit 1

What no. do you want to push in queue 11

What do you want to do??

1.Push 2.Pop 3.Size Of Queue 4.Display 0.Exit 1

What no. do you want to push in queue 12

What do you want to do??

1.Push 2.Pop 3.Size Of Queue 4.Display 0.Exit 1

What no. do you want to push in queue 13

What do you want to do??

1.Push 2.Pop 3.Size Of Queue 4.Display 0.Exit 3

Size of queue is 3

What do you want to do??

1.Push 2.Pop 3.Size Of Queue 4.Display 0.Exit 4

Queue is -->> 11 12 13

What do you want to do??

1. Push 2.Pop 3.Size Of Queue 4.Display 0.Exit 0

Process returned 0 (0x0) execution time : 28.868 s Press any key to continue.

/\* Practical No 08

Title: Write a program to create binary tree and perform operations on it.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/ #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(value)

{

struct node\* newNode = 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);

}

# OUTPUT:

Inorder traversal

5 ->12 ->6 ->1 ->9 ->

Preorder traversal

1 ->12 ->5 ->6 ->9 ->

Postorder traversal

5 ->6 ->12 ->9 ->1 ->

Process returned 0 (0x0) execution time : 0.087 s Press any key to continue.

/\* Practical No 09

Title: Write a program for creation of binary threaded tree.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/ #include<stdio.h> #include<malloc.h>

#define infinity 9999

typedef enum { thread,link} boolean; struct node \*in\_succ(struct node \*p); struct node \*in\_pred(struct node \*p); struct node

{

struct node \*left\_ptr; boolean left;

int info; boolean right;

struct node \*right\_ptr;

}\*head=NULL; int main()

{

int choice,num; insert\_head(); while(1)

{

printf("\n"); printf("1.Insert\n"); printf("2.Inorder Traversal\n"); printf("3.Quit\n"); printf("Enter your choice : "); scanf("%d",&choice); switch(choice)

{

case 1:

printf("Enter the number to be inserted : "); scanf("%d",&num);

insert(num); break;

case 2:

inorder(); break;

case 3:

exit(); default:

printf("Wrong choice\n");

}

}

}

int insert\_head()

{

struct node \*tmp;

head=(struct node \*)malloc(sizeof(struct node)); head->info= infinity;

head->left=thread; head->left\_ptr=head; head->right=link; head->right\_ptr=head;

}

int find(int item,struct node \*\*par,struct node \*\*loc)

{

struct node \*ptr,\*ptrsave; if(head->left\_ptr==head)

{

\*loc=NULL;

\*par=head; return;

}

if(item==head->left\_ptr->info)

{

\*loc=head->left\_ptr;

\*par=head; return;

}

ptr=head->left\_ptr; while(ptr!=head)

{

ptrsave=ptr;

if( item < ptr->info )

{

if(ptr->left==link) ptr=ptr->left\_ptr;

else

break;

}

else

if(item > ptr->info )

{

if(ptr->right==link) ptr=ptr->right\_ptr;

else

break;

}

if(item==ptr->info)

{

\*loc=ptr;

\*par=ptrsave; return;

}

}

\*loc=NULL;

\*par=ptrsave;

}

int insert(int item)

{

struct node \*tmp,\*parent,\*location; find(item,&parent,&location);

if(location!=NULL)

{

printf("Item already present"); return;

}

tmp=(struct node \*)malloc(sizeof(struct node)); tmp->info=item;

tmp->left=thread; tmp->right=thread;

if(parent==head) /\*tree is empty\*/

{

head->left=link; head->left\_ptr=tmp; tmp->left\_ptr=head; tmp->right\_ptr=head;

}

else

if( item < parent->info )

{

tmp->left\_ptr=parent->left\_ptr; tmp->right\_ptr=parent;

parent->left=link; parent->left\_ptr=tmp;

}

else

{

tmp->left\_ptr=parent;

tmp->right\_ptr=parent->right\_ptr; parent->right=link;

parent->right\_ptr=tmp;

}

}

struct node \*in\_succ(struct node \*ptr)

{

struct node \*succ; if(ptr->right==thread)

succ=ptr->right\_ptr; else

{

ptr=ptr->right\_ptr; while(ptr->left==link)

ptr=ptr->left\_ptr; succ=ptr;

}

return succ;

}

struct node \*in\_pred(struct node \*ptr)

{

struct node \*pred; if(ptr->left==thread)

pred=ptr->left\_ptr; else

{

ptr=ptr->left\_ptr; while(ptr->right==link)

ptr=ptr->right\_ptr; pred=ptr;

}

return pred;

}

inorder()

{

struct node \*ptr; if(head->left\_ptr==head)

{

printf("Tree is empty"); return;

}

ptr=head->left\_ptr; while(ptr->left==link)

ptr=ptr->left\_ptr; printf("%d ",ptr->info); while( 1 )

{

ptr=in\_succ(ptr); if(ptr==head)

break;

printf("%d ",ptr->info);

}

}

# OUTPUT:

Enter your choice : 1

Enter the number to be inserted : 2

1.Insert

1. Inorder Traversal 3.Quit

Enter your choice : 1

Enter the number to be inserted : 3

1.Insert

2.Inorder Traversal 3.Quit

Enter your choice : 1

Enter the number to be inserted : 4

1.Insert

2.Inorder Traversal 3.Quit

Enter your choice : 2 1 2 3 4

1.Insert

2.Inorder Traversal 3.Quit

Enter your choice : 3

/\* Practical No 10

Title: Write a program for depth first search and breadth first search.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

**//10.A C program to perform depth first search.**

#include<stdio.h> void DFS(int);

int G[10][10],visited[10],n; void main()

{

int i,j;

printf("Enter number of vertices:"); scanf("%d",&n);

printf("\nEnter adjecency matrix of the graph:"); for(i=0;i<n;i++)

for(j=0;j<n;j++)

scanf("%d",&G[i][j]); for(i=0;i<n;i++)

visited[i]=0; DFS(0);

}

void DFS(int i)

{

int j; printf("\n%d",i); visited[i]=1;

for(j=0;j<n;j++) if(!visited[j]&&G[i][j]==1)

DFS(j);

}

# OUTPUT:

Enter number of vertices:2

Enter adjecency matrix of the graph:10 12

14

18

0

Process returned 2 (0x2) execution time : 10.866 s Press any key to continue.

**//10.B: C program to perform breadth first search.**

#include<stdio.h>

int a[20][20], q[20], visited[20], n, i, j, f=0, r=-1; void bfs(int v)

{

for(i = 1; i <= n; i++) if(a[v][i] && !visited[i]) q[++r] = i;

if(f <= r)

{

visited[q[f]] = 1;

bfs(q[f++]);

}

}

void main()

{

int v;

printf("\n Enter the number of vertices:"); scanf("%d", &n);

for(i=1; i <= n; i++)

{

q[i] = 0;

visited[i] = 0;

}

printf("\n Enter graph data in matrix form:\n"); for(i=1; i<=n; i++)

{

for(j=1;j<=n;j++)

{

scanf("%d", &a[i][j]);

}

}

printf("\n Enter the starting vertex:"); scanf("%d", &v);

bfs(v);

printf("\n The node which are reachable are:\n"); for(i=1; i <= n; i++)

{

if(visited[i]) printf("%d\t", i);

else

{

printf("\n Bfs is not possible. Not all nodes are reachable"); break;

}

}

}

# OUTPUT:

Enter the number of vertices:3

Enter graph data in matrix form: 1 2 3

4 5 6

7 8 9

Enter the starting vertex:1

The node which are reachable are: 1 2 3

/\*

Practical No 11

Title: Write a program for bubble sort and bucket sort.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

// C program to perform Bubble sort. #include <stdio.h>

void swap(int \*xp, int \*yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void bubbleSort(int arr[], int n)

{

int i, j; for(i=0;i<n-1;i++)

for(j=0;j<n-i-1;j++)

if (arr[j]>arr[j+1])

swap(&arr[j], &arr[j+1]);

}

void printArray(int arr[], int size)

{

int i; for(i=0;i<size;i++)

printf("%d ", arr[i]); printf("\n");

}

int main()

{

int arr[] = {64, 34, 25, 12, 22, 11, 90};

int n=sizeof(arr)/sizeof(arr[0]); bubbleSort(arr, n); printf("Sorted array: \n"); printArray(arr, n);

return 0;

}

OUTPUT:

Sorted array:

11 12 22 25 34 64 90

Process returned 0 (0x0) execution time : 0.033 s Press any key to continue.

**//11.B C program to perform bucket sort.**

#include <stdio.h>

void Bucket\_Sort(int array[], int n)

{

int i, j;

int count[n]; for(i=0;i<n;i++)

count[i]=0; for(i=0;i<n;i++)

(count[array[i]])++; for(i=0 ,j=0;i<n;i++)

for(;count[i]>0;(count[i])--) array[j++] = i;

}

int main()

{

int array[100], i, num;

printf("Enter the size of array : "); scanf("%d", &num);

printf("Enter the %d elements to be sorted:\n",num); for(i=0;i<num;i++)

scanf("%d", &array[i]);

printf("\nThe array of elements before sorting : \n"); for(i=0;i<num;i++)

printf("%d ", array[i]);

printf("\nThe array of elements after sorting : \n"); Bucket\_Sort(array, num);

for(i=0;i<num;i++) printf("%d ", array[i]);

printf("\n"); return 0;

}

# OUTPUT:

Enter the size of array : 10

Enter the 10 elements to be sorted:

2 3 5 1 6 4 7 3 8 7

The array of elements before sorting : 2 3 5 1 6 4 7 3 8 7

The array of elements after sorting : 1 2 3 3 4 5 6 7 7 8

Process returned 0 (0x0) execution time : 19.412 s Press any key to continue.

/\*

Practical No 12

Title: Write a program for merge sort and heap sort.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

**//12.A C program to perform merge sort.**

#include <stdio.h> #define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };

int b[10];

void merging(int low, int mid, int high)

{

int l1, l2, i;

for(l1 = low,l2=mid+1,i=low;l1<=mid&&l2<=high;i++)

{

if(a[l1]<=a[l2])

b[i]=a[l1++]; else

b[i]=a[l2++];

}

while(l1<=mid) b[i++]=a[l1++];

while(l2<=high) b[i++]=a[l2++];

for(i=low;i<=high;i++) a[i]=b[i];

}

void sort(int low, int high)

{

int mid; if(low<high)

{

mid=(low+high)/2; sort(low, mid); sort(mid+1, high); merging(low, mid, high);

}

else

{

return;

}

}

int main()

{

int i;

printf("List before sorting\n"); for(i=0;i<=max;i++)

printf("%d ", a[i]); sort(0, max);

printf("\n\nList after sorting\n"); for(i=0;i<=max;i++)

printf("%d ", a[i]);

}

# OUTPUT:

List before sorting

10 14 19 26 27 31 33 35 42 44 0

List after sorting

0 10 14 19 26 27 31 33 35 42 44

### //12.B C program to perform heap sort.

#include <stdio.h> void main()

{

int heap[10], no, i, j, c, root, temp; printf("Enter no of elements:"); scanf("%d", &no);

printf("\nEnter the nos: "); for(i=0;i<no;i++)

scanf("%d", &heap[i]); for(i=1;i<no;i++)

{

c=i; do

{

root=(c-1)/2; if(heap[root]<heap[c])

{

temp=heap[root]; heap[root]=heap[c]; heap[c]=temp;

}

c=root;

}while(c!=0);

}

printf("Heap array: "); for(i=0;i<no;i++)

printf("%d\t", heap[i]);

for(j=no-1;j>=0;j--)

{

temp=heap[0]; heap[0]=heap[j]; heap[j]=temp; root=0;

do

{

c=2\*root+1;

if((heap[c]<heap[c + 1])&&c<j-1) c++;

if(heap[root]<heap[c]&&c<j)

{

temp=heap[root]; heap[root]=heap[c]; heap[c]=temp;

}

root=c;

}while(c<j);

}

printf("\n\nThe sorted array is:"); for(i=0;i<no;i++)

printf("%d\t", heap[i]);

printf("\nComplexity: Best case= Avg case= Worst case= O(n logn)\n");

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **OUTPUT:** |  | | | |
| Enter no of elements:5 |  |  |  |  |
| Enter the nos: 14 12 13 15  Heap array: 15 14 13 | 11 | 12 | 11 |  |
| The sorted array is:11 12 |  | 13 | 14 | 15 |

Complexity: Best case= Avg case= Worst case= O(n logn)

Process returned 0 (0x0) execution time : 7.815 s Press any key to continue.

/\*

Practical No 13

Title: Write a program for insertion sort and quick sort.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

### //13.A C program to perform insertion sort.

#include <stdio.h> int main()

{

int n, array[1000], c, d, t, flag = 0; printf("Enter number of elements\n"); scanf("%d", &n);

printf("Enter %d integers\n", n); for(c=0;c<n;c++)

scanf("%d", &array[c]); for(c=1;c<=n-1;c++)

{

t=array[c];

for(d=c-1;d>=0;d--)

{

if(array[d]>t)

{

array[d+1]=array[d]; flag=1;

}

else

break;

}

if (flag) array[d+1]=t;

}

printf("Sorted list in ascending order:\n"); for(c=0;c<=n-1;c++)

{

printf("%d\n", array[c]);

}

return 0;

}

# OUTPUT:

Enter number of elements 5

Enter 5 integers

11 14 12 15 12

Sorted list in ascending order:

11

12

12

14

15

Process returned 0 (0x0) execution time : 9.063 s Press any key to continue.

**//13.B C program to perform quick sort.**

#include <stdio.h>

void quick\_sort(int[],int,int); int partition(int[],int,int); int main()

{

int a[50],n,i;

printf("Enter the number of elements:"); scanf("%d",&n);

printf("\nEnter array elements:"); for(i=0;i<n;i++)

scanf("%d",&a[i]); quick\_sort(a,0,n-1); printf("\nArray after sorting:"); for(i=0;i<n;i++)

printf("%d ",a[i]); return 0;

}

void quick\_sort(int a[],int l,int u)

{

int j; if(l<u)

{

j=partition(a,l,u); quick\_sort(a,l,j-1); quick\_sort(a,j+1,u);

}

}

int partition(int a[],int l,int u)

{

int v,i,j,temp; v=a[l];

i=l; j=u+1; do

{

do

i++;

while(a[i]<v&&i<=u); do

j--;

while(v<a[j]); if(i<j)

{

temp=a[i]; a[i]=a[j]; a[j]=temp;

}

}while(i<j);

a[l]=a[j]; a[j]=v; return(j);

}

# OUTPUT:

Enter the number of elements:5 Enter array elements:4 2 3 5 1

Array after sorting:1 2 3 4 5

/\*

Practical No 14

Title: Write a program for binary search.

Name of Programmer: Vishal Pravin Jatti

Roll No: MC23F14F026

\*/

#include <stdio.h> int main()

{

int c, first, last, middle, n, search, array[100]; printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n); for(c=0;c<n;c++)

scanf("%d", &array[c]); printf("Enter value to find\n"); scanf("%d", &search);

first=0; last=n-1;

middle=(first+last)/2; while(first<=last)

{

if(array[middle]<search) first=middle+1;

else if(array[middle]==search)

{

printf("%d found at location %d.\n", search, middle+1); break;

}

else

last=middle-1; middle=(first+last)/2;

}

if(first>last)

printf("Not found! %d isn't present in the list.\n", search); return 0;

}

# OUTPUT:

Enter number of elements 5

Enter 5 integers

11

12

13

14

15

Enter value to find 14

14 found at location 4.

Process returned 0 (0x0) execution time : 8.998 s Press any key to continue.