**DSA PRACTICAL FILE**

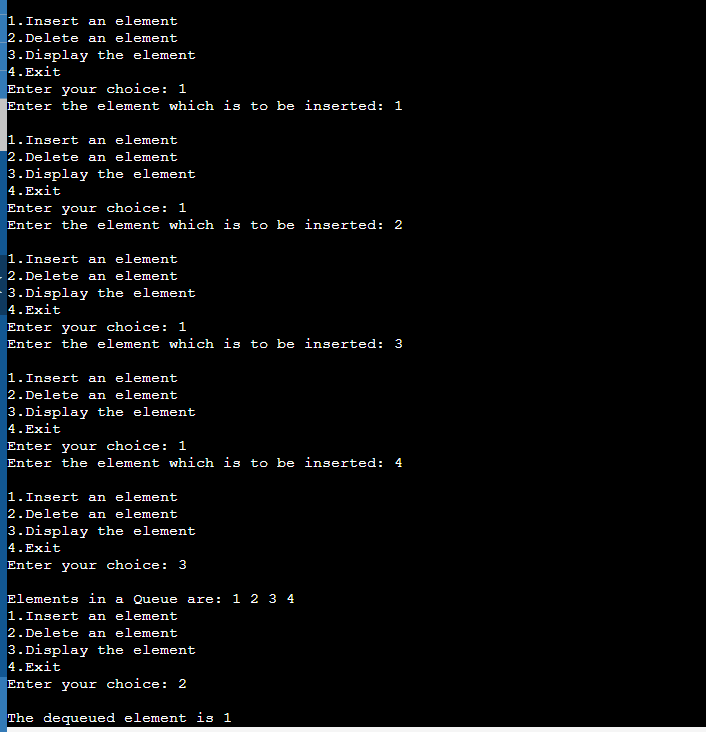
(CodeChef & online GDB as C language IDE is used for all the programs)

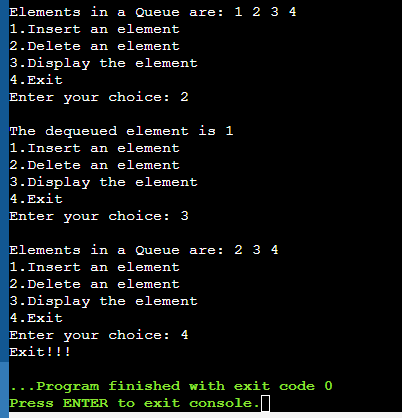
Practical File programs from 17 to 20

Name: Hardik Chhabra

ECAM-1

Roll no.: 2020UEA6504

**OUTPUT-17**

****

**Practical-17**

**Q. Write a program to implement a circular queue using an array.**

#include <stdio.h>

# define max 6

int queue[max];

int front=-1;

int rear=-1;

void enqueue(int element) {

if(front==-1 && rear==-1) {

front=0;

rear=0;

queue[rear]=element;

}else if((rear+1)%max==front) {

printf("Queue is overflow..");

}else {

rear=(rear+1)%max;

queue[rear]=element;

}

}

int dequeue() {

if((front==-1)&&(rear==-1)) {

printf("\nUNDERFLOW!!!");

}else if(front==rear) {

printf("\nThe dequeued element is %d", queue[front]);

front=-1;

rear=-1;

}else {

printf("\nThe dequeued element is %d", queue[front]);

front=(front+1)%max;

}

}

void display() {

int i=front;

if(front==-1 && rear==-1) {

printf("\nQueue is empty!!!");

}else {

printf("\nElements in a Queue are:");

while(i<=rear) {

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

i=(i+1)%max;

}

}

}

int main(void) {

int choice=1,x;

while(choice<4 && choice!=0) {

printf("\n1.Insert an element");

printf("\n2.Delete an element");

printf("\n3.Display the element");

printf("\n4.Exit");

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1: printf("Enter the element which is to be inserted");

scanf("%d", &x);

enqueue(x);

break;

case 2: dequeue();

break;

case 3: display();

break;

case 4: printf("Exit!!!");

exit(0);

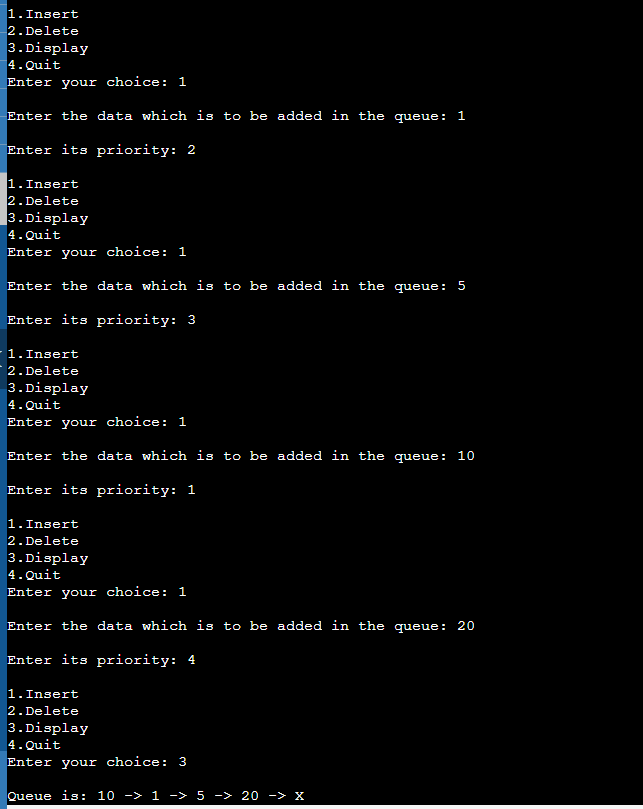
}

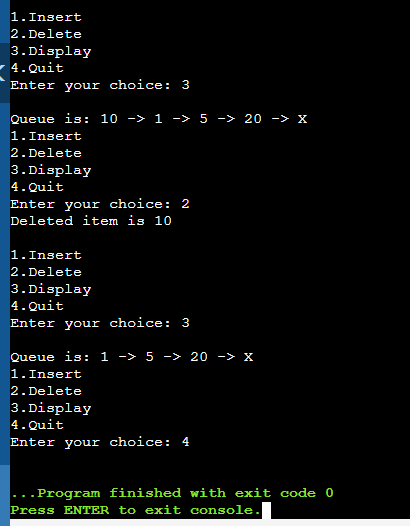
}

return 0;

}

**OUTPUT-18**





**Practical-18**

**Q. Write a program to implement a priority queue using a linked list.**

# include<stdio.h>

# include<malloc.h>

typedef struct node {

int priority;

int info;

struct node \*link;

}NODE;

NODE \*front = NULL;

void insert(int data,int priority) {

NODE \*temp,\*q;

temp = (NODE \*)malloc(sizeof(NODE));

temp->info = data;

temp->priority = priority;

if( front == NULL || priority < front->priority ) {

temp->link = front;

front = temp;

}else {

q = front;

while( q->link != NULL && q->link->priority <= priority)

q=q->link;

temp->link = q->link;

q->link = temp;

}

}

void del() {

NODE \*temp;

if(front == NULL)

printf("UNDERFLOW!!!\n");

else {

temp = front;

printf("Deleted item is %d\n", temp->info);

front = front->link;

free(temp);

}

}

void display() {

NODE \*ptr;

ptr = front;

if(front == NULL)

printf("Queue is empty!!!\n");

else {

printf("\nQueue is:");

//printf("Priority Item\n");

while(ptr != NULL) {

//printf("%5d %5d\n",ptr->priority,ptr->info);

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

ptr = ptr->link;

}

printf(" X");

}

}

int main(void) {

int choice, data, priority;

do {

printf("\n1.Insert");

printf("\n2.Delete");

printf("\n3.Display");

printf("\n4.Quit");

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1: printf("\nEnter the data which is to be added in the queue: ");

scanf("%d",&data);

printf("\nEnter its priority: ");

scanf("%d",&priority);

insert(data,priority);

break;

case 2: del();

break;

case 3: display();

break;

case 4: break;

default : printf("\nWrong choice!!!");

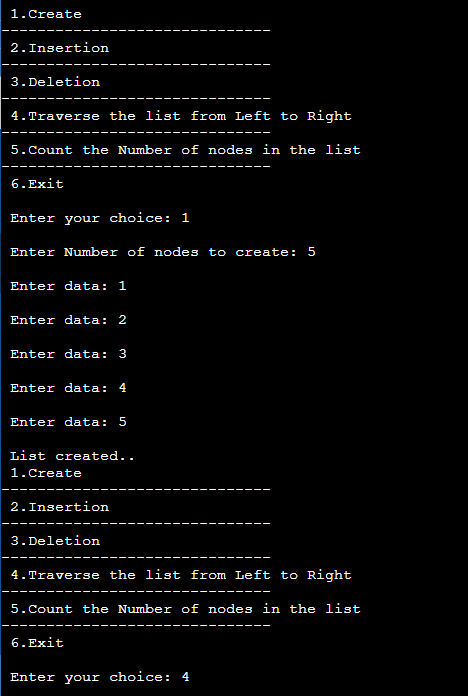
}

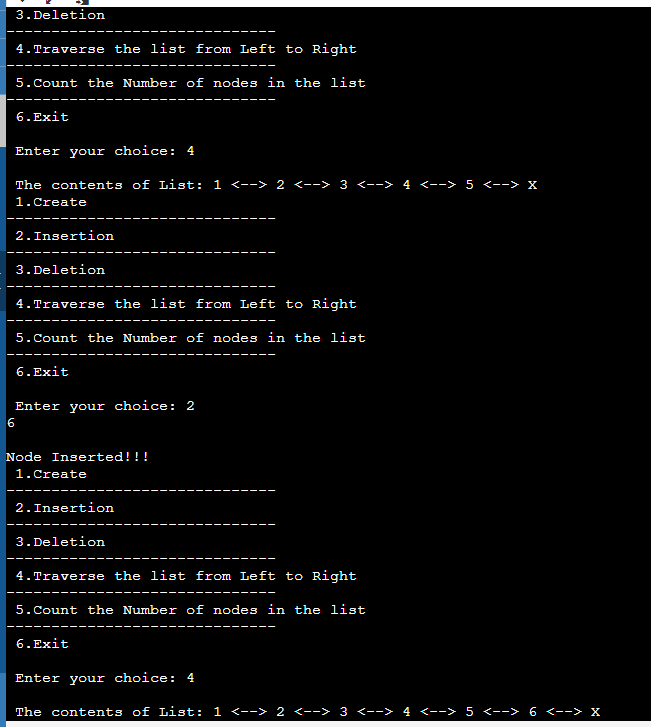
}while(choice!=4);

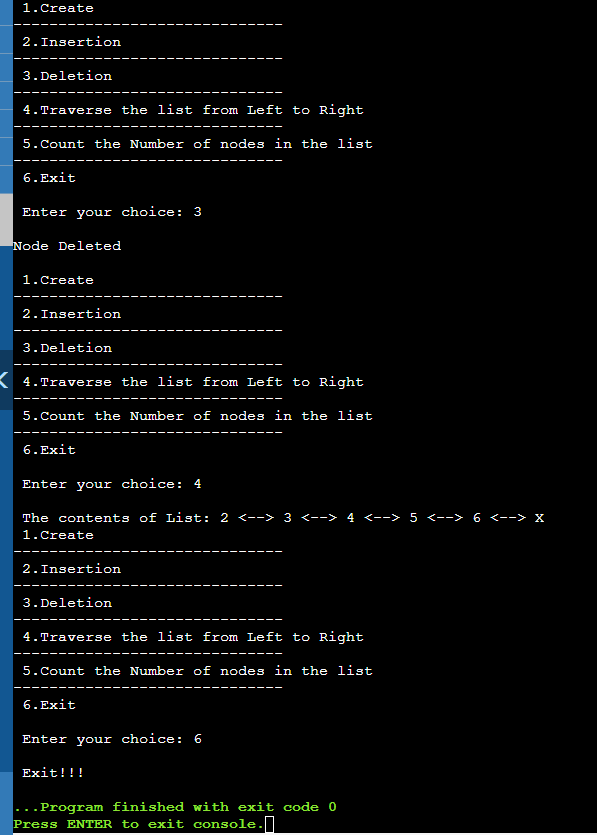
return 0;

}

**OUTPUT-19**

****

****

****

**Practical-19**

**Q. Write a program to implement a double ended queue using a linked list.**

#include <stdio.h>

#include <stdlib.h>

struct dlinklist {

struct dlinklist \*left;

int data;

struct dlinklist \*right;

};

typedef struct dlinklist node;

node \*start = NULL;

node\* getnode() {

node \* newnode;

newnode = (node \*) malloc(sizeof(node));

printf("\n Enter data: ");

scanf("%d", &newnode -> data);

newnode -> left = NULL;

newnode -> right = NULL;

return newnode;

}

int countnode(node \*start) {

if(start == NULL) {return 0;}

else {return (1 + countnode(start -> right));}

}

int menu() {

int ch;

printf("\n 1.Create");

printf("\n------------------------------");

printf("\n 2.Insertion");

printf("\n------------------------------");

printf("\n 3.Deletion");

printf("\n------------------------------");

printf("\n 4.Traverse the list from Left to Right");

printf("\n------------------------------");

printf("\n 5.Count the Number of nodes in the list");

printf("\n------------------------------");

printf("\n 6.Exit");

printf("\n\n Enter your choice: ");

scanf("%d", &ch);

return ch;

}

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 -> right)

{temp = temp -> right;}

temp -> right = newnode;

newnode -> left = temp;

}

}

}

void ins(int item) {

node \*ptr = (node \*) malloc(sizeof(node));

node \*temp;

if(ptr == NULL) {

printf("\nOVERFLOW");

}else {

ptr->data=item;

if(start == NULL) {

ptr->right = NULL;

ptr->left = NULL;

start = ptr;

}else {

temp = start;

while(temp->right!=NULL) {

temp = temp->right;

}

temp->right = ptr;

ptr ->left=temp;

ptr->right = NULL;

}

printf("\nNode Inserted!!!");

}

}

void del() {

node \*ptr;

if(start == NULL) {

printf("\n UNDERFLOW\n");

}else if(start->right == NULL) {

start = NULL;

free(start);

printf("\nNode Deleted\n");

}else {

ptr = start;

start = start -> right;

start -> left = NULL;

free(ptr);

printf("\nNode Deleted\n");

}

}

void traverse() {

node \*temp;

temp = start;

printf("\n The contents of List: ");

if(start == NULL )

{printf("\n Empty List");}

else {

while(temp != NULL) {

printf("%d <--> ", temp -> data);

temp = temp -> right;

}

printf("X");

}

}

int main(void) {

int ch, n,prod;

while(1) {

ch = menu();

switch(ch) {

case 1: printf("\n Enter Number of nodes to create: ");

scanf("%d", &n);

createlist(n);

printf("\n List created..");

break;

case 2: scanf("%d",&prod);

ins(prod);

break;

case 3: del();

break;

case 4: traverse();

break;

case 5: printf("\n Number of nodes: %d", countnode(start));

break;

case 6: printf("\n Exit!!!");

exit(0);

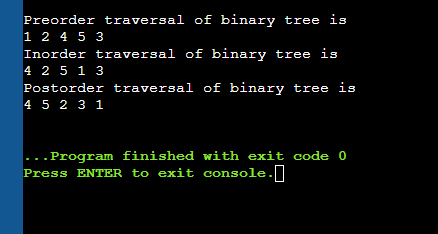
}

}

return 0;

}

**OUTPUT-20**

****

**Practical-20**

**Q. Write a program to construct a binary tree and display its pre-order, in-order and post-order traversals.**

#include <stdio.h>

#include <stdlib.h>

struct node {

int data;

struct node\* left;

struct node\* right;

};

struct node\* newNode(int data)

{

struct node\* node

= (struct node\*)malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return (node);

}

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

printPostorder(node->left);

printPostorder(node->right);

printf("%d ", node->data);

}

void printInorder(struct node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

printf("%d ", node->data);

printInorder(node->right);

}

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

printf("%d ", node->data);

printPreorder(node->left);

printPreorder(node->right);

}

int main()

{

struct node\* root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

printf("\nPreorder traversal of binary tree is \n");

printPreorder(root);

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

printf("\nPostorder traversal of binary tree is \n");

printPostorder(root);

getchar();

return 0;

}