**Q1.Implementation of Stack using Array.**

**#include<iostream.h>**

**#include<conio.h>**

**int stack[20],n=20,top=-1;**

**void push(int val)**

**{**

**if(top>=n-1)**

**cout<<"Stack Overflow!!"<<endl;**

**else {**

**top++;**

**stack[top]=val;**

**}**

**}**

**void pop()**

**{**

**if(top<=-1)**

**cout<<"Stack Underflow!!"<<endl;**

**else {**

**cout<<"The Popped element is: "<<stack[top]<<endl;**

**top--;**

**}**

**}**

**void display()**

**{**

**if(top>=0){**

**cout<<"Stack elements are: ";**

**for(int i=top;i>=0;i--)**

**cout<<stack[i]<<" ";**

**cout<<endl;**

**}**

**else**

**cout<<"Stack is empty";**

**}**

**int main()**

**{**

**int ch,val;**

**cout<<"1. Push in Stack"<<endl;**

**cout<<"2. Pop from Stack"<<endl;**

**cout<<"3. Display Stack"<<endl;**

**cout<<"4. Exit"<<endl;**

**do{**

**cout<<"\nEnter choice : "<<endl;**

**cin>>ch;**

**switch(ch) {**

**case 1:{**

**cout<<"Enter value to be pushed: "<<endl;**

**cin>>val;**

**push(val);**

**break;**

**}**

**case 2: {**

**pop();**

**break;**

**}**

**case 3: {**

**display();**

**break;**

**}**

**case 4: {**

**cout<<"Exit!"<<endl;**

**break;**

**}**

**default: {**

**cout<<"Invalid Choice"<<endl;**

**}**

**}**

**} while(ch!=4);**

**getch();**

**return 0;**

**}**

**Q2. Implementation of Queue using Array.**

**#include<iostream.h>**

**#include<conio.h>**

**int queue[20],n=20,front=-1,rear=-1;**

**void Insert()**

**{**

**int val;**

**if(rear==n-1)**

**cout<<"Queue Overflow!!"<<endl;**

**else{**

**if(front==-1)**

**front=0;**

**cout<<"Insert the element in the Queue: "<<endl;**

**cin>>val;**

**rear++;**

**queue[rear]=val;**

**}**

**}**

**void Delete()**

**{**

**if(front==-1||front>rear)**

**{**

**cout<<"Queue Underflow!!";**

**return;**

**}**

**else{**

**cout<<"Element deleted from the queue is: "<<queue[front]<<endl;**

**front++;**

**}**

**}**

**void display()**

**{**

**if(front==-1)**

**cout<<"Queue is empty"<<endl;**

**else{**

**cout<<"Queue elements are: ";**

**for(int i=front;i<=rear;i++)**

**cout<<queue[i]<<" ";**

**cout<<endl;**

**}**

**}**

**int main()**

**{**

**clrscr();**

**int ch;**

**cout<<"1.Insert element to queue"<<endl;**

**cout<<"2.Delete element from queue"<<endl;**

**cout<<"3.Display all the elements of queue"<<endl;**

**cout<<"4.Exit"<<endl;**

**do{**

**cout<<"Enter your choice : "<<endl;**

**cin>>ch;**

**switch(ch){**

**case 1: Insert();**

**break;**

**case 2: Delete();**

**break;**

**case 3: display();**

**break;**

**case 4: cout<<"Exit!"<<endl;**

**break;**

**default: cout<<"Invalid Choice!!";**

**}**

**} while(ch!=4);**

**getch();**

**return 0;**

**}**

**Q3. Implementation of Circular Queue using Array.**

**#include<iostream.h>**

**#include<conio.h>**

**int cqueue[5];**

**int front=-1,rear=-1,n=5;**

**void Insert(int val) {**

**if((front==0&&rear==n-1) ||(front==rear+1)){**

**cout<<"Circular Queue Overflow!!\n";**

**return;**

**}**

**if(front==-1){**

**front=0;**

**rear=0;**

**}else{**

**if(rear==n-1)**

**rear=0;**

**else**

**rear=rear+1;**

**}**

**cqueue[rear]=val;**

**}**

**void Delete(){**

**if (front==-1) {**

**cout<<"Circular Queue Underflow\n";**

**return;**

**}**

**cout<<"Element deleted from queue is:"<<cqueue[front]<<endl;**

**if(front==rear) {**

**front=-1;**

**rear=-1;**

**}else{**

**if(front==n-1)**

**front=0;**

**else**

**front=front+1;**

**}}**

**void display(){**

**int f=front,r=rear;**

**if(front==-1) {**

**cout<<"Queue is empty"<<endl;**

**return;**

**}**

**cout<<"Queue elements are: \n";**

**if(f<=r) {**

**while(f<=r) {**

**cout<<cqueue[f]<<" ";**

**f++;**

**}**

**}else{**

**while(f<=n-1) {**

**cout<<cqueue[f]<<" ";**

**f++;**

**}**

**f=0;**

**while(f<=r) {**

**cout<<cqueue[f]<<" ";**

**f++;**

**}}**

**cout<<endl;**

**}**

**int main(){**

**int ch,val;**

**clrscr();**

**cout<<"1. Insert\n";**

**cout<<"2. Delete\n";**

**cout<<"3. Display\n";**

**cout<<"4.Exit\n";**

**do{**

**cout<<"Enter Choice: "<<endl;**

**cin>>ch;**

**switch(ch){**

**case 1:**

**cout<<"Input for Insertion:"<<endl;**

**cin>>val;**

**Insert(val);**

**break;**

**case 2:**

**Delete();**

**break;**

**case 3:**

**display();**

**break;**

**case4:**

**cout<<"Exit\n";**

**break;**

**default: cout<<"Incorrect!!\n";**

**}**

**}while(ch!=4);**

**getch();**

**return 0;**

**}**

**Q4. WAP to implement a Link-List .**

**#include<iostream.h>**

**#include<conio.h>**

**#include<alloc.h>**

**struct node**

**{**

**int data;**

**struct node \*next;**

**};**

**struct node \*ptr,\*head;**

**void add\_at\_begin()**

**{**

**struct node \*temp;**

**int item;**

**cout<<"\nenter item to add at begining";**

**cin>>item;**

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

**temp->data=item;**

**temp->next=head;**

**head=temp;**

**}**

**void add\_at\_pos()**

**{**

**struct node \*temp,\*ptr,\*old;**

**int item,pos,i;**

**cout<<"\nEnter item to add at begining";**

**cin>>item;**

**cout<<"\nenter position";**

**cin>>pos;**

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

**temp->data=item;**

**ptr=head;**

**i=1;**

**while(i<pos&&ptr!=NULL)**

**{**

**old=ptr;**

**ptr=ptr->next;**

**i++;**

**}**

**if(i>pos)**

**cout<<"position is greater than elemenerts";**

**if(ptr==NULL)**

**cout<<"\nposition is not valid";**

**if(i==pos && ptr!=NULL)**

**{**

**temp->next=ptr;**

**old->next=temp;**

**}**

**}**

**void add\_at\_end()**

**{**

**struct node \*ptr,\*temp;**

**int item;**

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

**cout<<"\nEnter data to insert";**

**cin>>item;**

**temp->data=item;**

**temp->next=NULL;**

**if(head==NULL)**

**{**

**head=temp;**

**}**

**else**

**{**

**ptr=head;**

**while(ptr->next!=NULL)**

**ptr=ptr->next;**

**ptr->next=temp;**

**}**

**cout<<"\n Data is inserted";**

**}**

**void show\_all()**

**{**

**struct node \*ptr;**

**ptr=head;**

**while(ptr!=NULL)**

**{**

**cout<<ptr->data<<"->";**

**ptr=ptr->next;**

**}**

**}**

**void find()**

**{**

**struct node \*ptr;**

**int item,i;**

**cout<<"\nenter no to find ";**

**cin>>item;**

**ptr=head;**

**while(ptr->data!=item)**

**{**

**if(ptr==NULL)**

**{**

**cout<<"\nData not found";**

**return;}**

**ptr=ptr->next;**

**}**

**if(ptr->data==item)**

**cout<<"\nData found";**

**}**

**void count()**

**{**

**struct node \*ptr;**

**ptr=head;**

**int i=0;**

**while(ptr!=NULL)**

**{**

**i++;**

**ptr=ptr->next;}**

**cout<<"\nNumber of nodes="<<i;**

**}**

**void delete\_at\_begin()**

**{**

**struct node \*temp;**

**temp=head;**

**if(temp==NULL)**

**{**

**cout<<"\nLink list is empty";**

**return;**

**}**

**head=head->next;**

**cout<<"\nFirst node is detected";**

**free(temp);**

**}**

**void delete\_at\_end()**

**{**

**struct node \*ptr, \*old;**

**ptr=head;**

**if(ptr==NULL)**

**{**

**cout<<"\nList is emty";**

**return;**

**}**

**while(ptr->next!=NULL)**

**{**

**old=ptr;**

**ptr=ptr->next;**

**}**

**old->next=NULL;**

**cout<<"\nDeleted Data="<<ptr->data;**

**free(ptr);**

**}**

**void delete\_at\_pos()**

**{**

**struct node \*ptr, \*old;**

**int pos,i;**

**//if(temp==NULL)**

**cout<<"\Enter position:";**

**cin>>pos;**

**ptr=head;**

**if(ptr==NULL)**

**{**

**cout<<"\nList is Empty";**

**return;**

**}**

**i=1;**

**while(i<pos)**

**{**

**if(ptr==NULL)**

**{**

**cout<<"\nLess elements than position";**

**return;**

**}**

**old=ptr;**

**ptr=ptr->next;**

**i++;**

**}**

**old->next=ptr->next;**

**cout<<"\nDeleted data:"<<ptr->data;**

**free(ptr);**

**}**

**void main()**

**{**

**int choice;**

**head=NULL;**

**do**

**{**

**clrscr();**

**cout<<"\n Main Menu for Link list";**

**cout<<"\n1. Add new element at end";**

**cout<<"\n2. Add new element at beginning";**

**cout<<"\n3. Add new element at given position";**

**cout<<"\n4. Show all data";**

**cout<<"\n5. Delete data from beginning";**

**cout<<"\n6. Delete data from end";**

**cout<<"\n7. Delete data from given position";**

**cout<<"\n8. Count the data";**

**cout<<"\n9. Find the data";**

**cout<<"\n10. Exit";**

**cout<<"\n Enter Your choice :";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**add\_at\_end();**

**break;**

**case 2:**

**add\_at\_begin();**

**break;**

**case 3:**

**add\_at\_pos();**

**break;**

**case 4:**

**show\_all();**

**break;**

**case 5:**

**delete\_at\_begin();**

**break;**

**case 6:**

**delete\_at\_end();**

**break;**

**case 7:**

**delete\_at\_pos();**

**break;**

**case 8:**

**count();**

**case 9:**

**find();**

**break;**

**case 10:**

**cout<<"\n Thank you";**

**break;**

**default:**

**cout<<"\n Wrong choice!!! \n Try Again";**

**}**

**getch();**

**}while(choice!=10);**

**}**

**Q8.WAP to show working of Selection Sort.**

**#include<iostream.h>**

**#include<conio.h>**

**void selection\_sort(int a[],int size)**

**{**

**int small,temp,pos;**

**for(int i=0;i<size-1;i++)**

**{**

**small=a[i];**

**pos=i;**

**for(int j=i+1;j<size;j++)**

**{**

**if(small>a[j])**

**{**

**small=a[j];**

**pos=j;**

**}**

**}**

**if(pos!=i)**

**{**

**temp=a[i];**

**a[i]=small;**

**a[pos]=temp;**

**}**

**}**

**}**

**int main()**

**{**

**clrscr();**

**int i,s,b[10];**

**cout<<"Enter the size of Array";**

**cin>>s;**

**cout<<"\nEnter the elements of array";**

**for(i=0;i<s;i++)**

**{**

**cin>>b[i];**

**}**

**selection\_sort(b,s);**

**for(i=0;i<s;i++)**

**{**

**cout<<b[i]<<"\t";**

**}**

**getch();**

**return 0;**

**}**

**Q9.WAP to show working of Bubble Sort.**

#include<iostream.h>

#include<conio.h>

void bubble\_sort(int a[],int size)

{

int temp;

for(int i=0;i<size-1;i++)

{

for(int j=0;j<size-1;j++)

{

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

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

}

int main()

{

clrscr();

int I,s,b[10];

cout<<”Enter the size of Array”;

cin>>s;

cout<<”\nEnter the elements of array”;

for(i=0;i<s;i++)

{

cin>>b[i];

}

bubble\_sort(b,s);

for(i=0;i<s;i++)

{

cout<<b[i]<<”\t”;

}

getch();

return 0;

}

**Q10.Wap to show working of Insertion Sort.**

**#include<iostream.h>**

**#include<conio.h>**

**void insertion\_sort(int a[],int size)**

**{**

**int key,j;**

**for(int i=1;i<size;i++)**

**{**

**key=a[i];**

**j=i-1;**

**while(j>=0 && key<a[j])**

**{**

**a[j+1]=a[j];**

**j--;**

**}**

**a[j+1]=key;**

**}**

**}**

**int main()**

**{**

**clrscr();**

**int i,s,b[10];**

**cout<<"Enter the size of Array";**

**cin>>s;**

**cout<<"\nEnter the elements of array";**

**for(i=0;i<s;i++)**

**{**

**cin>>b[i];**

**}**

**insertion\_sort(b,s);**

**for(i=0;i<s;i++)**

**{**

**cout<<b[i]<<"\t";**

**}**

**getch();**

**return 0;**

**}**

**Q12. WAP to show working of Heap Sort.**

**#include <iostream.h>**

**#include<conio.h>**

**void heapify(int arr[], int n, int i)**

**{**

**int largest = i;**

**int l = 2 \* i + 1;**

**int r = 2 \* i + 2**

**if (l < n && arr[l] > arr[largest])**

**largest = l;**

**if (r < n && arr[r] > arr[largest])**

**largest = r;**

**if (largest != i) {**

**int temp;**

**temp=arr[i];**

**arr[i]=arr[largest];**

**arr[largest]=temp;**

**heapify(arr, n, largest);**

**}**

**}**

**void heapSort(int arr[], int n)**

**{**

**for (int i = n / 2 - 1; i >= 0; i--)**

**heapify(arr, n, i);**

**for (i = n - 1; i > 0; i--) {**

**int c;**

**c=arr[0];**

**arr[0]=arr[i];**

**arr[i]=c;**

**heapify(arr, i, 0);**

**}**

**}**

**void printArray(int arr[], int n)**

**{**

**for (int i = 0; i < n; ++i)**

**cout << arr[i] << " ";**

**cout << "\n";**

**}**

**int main()**

**{ clrscr();**

**int n;**

**int arr[10];**

**cout<<"\nEnter the size of Array:";**

**cin>>n;**

**cout<<"\nEnter the Array elements:";**

**for(int i=0;i<n;i++)**

**{**

**cin>>arr[i];**

**}**

**heapSort(arr, n);**

**cout << "Sorted array is \n";**

**printArray(arr, n);**

**getch();**

**return 0;**

**}**

**11.WAP to show working of Merge Sort.**

**#include<iostream.h>**

**#include<conio.h>**

**void swapping(int &a, int &b) {**

**int temp;**

**temp = a;**

**a = b;**

**b = temp;**

**}**

**void display(int \*array, int size) {**

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

**cout << array[i] << " ";**

**cout << endl;**

**}**

**void merge(int \*array, int l, int m, int r) {**

**int i, j, k, nl, nr;**

**nl = m-l+1; nr = r-m;**

**int larr[10], rarr[10];**

**for(i = 0; i<nl; i++)**

**larr[i] = array[l+i];**

**for(j = 0; j<nr; j++)**

**rarr[j] = array[m+1+j];**

**i = 0; j = 0; k = l;**

**while(i < nl && j<nr) {**

**if(larr[i] <= rarr[j]) {**

**array[k] = larr[i];**

**i++;**

**}else{**

**array[k] = rarr[j];**

**j++;**

**}**

**k++;**

**}**

**while(i<nl)**

**array[k] = larr[i];**

**i++; k++;**

**}**

**while(j<nr) {**

**array[k] = rarr[j];**

**j++; k++;**

**}**

**}**

**void mergeSort(int \*array, int l, int r) {**

**int m;**

**if(l < r) {**

**int m = l+(r-l)/2;**

**mergeSort(array, l, m);**

**mergeSort(array, m+1, r);**

**merge(array, l, m, r);**

**}**

**}**

**int main() {**

**int n;**

**clrscr();**

**cout << "Enter the number of elements: ";**

**cin >> n;**

**int arr[10];**

**cout << "Enter elements:" << endl;**

**for(int i = 0; i<n; i++) {**

**cin >> arr[i];**

**}**

**cout << "Array before Sorting: ";**

**display(arr, n);**

**mergeSort(arr, 0, n-1);**

**cout << "Array after Sorting: ";**

**display(arr, n);**

**getch();**

**return 0;**

**}**

**Q13. WAP to show working of Linear Search.**

**#include<iostream.h>**

**#include<conio.h>**

**int main()**

**{**

**int arr[10],n=10,i;**

**clrscr();**

**cout<<"\nEnter Array Values:";**

**for(i=0;i<n;i++)**

**cin>>arr[i];**

**cout<<"\nEnter the number you want to search:";**

**cin>>n;**

**int pos=-1;**

**for(i=0;i<10;i++)**

**{**

**if(n==arr[i])**

**{**

**pos=i;**

**break;**

**}**

**}**

**if(pos==-1)**

**{**

**cout<<n<<" is not in the Array!";**

**}**

**else**

**{**

**cout<<"\nElement found at "<<pos+1<<" position";**

**}**

**getch();**

**return 0;**

**}**

**Q14. WAP to show working of Binary Search.**

**#include<iostream.h>**

**#include<conio.h>**

**int main()**

**{**

**int a[10],n,i,lb,ub,mid=0,pos=-1;**

**clrscr();**

**cout<<"\nEnter Array Values!";**

**for(i=0;i<10;i++)**

**cin>>a[i];**

**cout<<"Enter the number you want to search:";**

**cin>>n;**

**lb=0;**

**ub=9;**

**while((lb<ub)&&(a[mid]!=n))**

**{**

**mid=(lb+ub)/2;**

**if(a[mid]==n)**

**{ pos=mid;**

**break;**

**}**

**else if(a[mid]>n)**

**lb=mid-1;**

**else**

**lb=mid+1;**

**}**

**if(pos==-1)**

**cout<<"\nNumber is not found!";**

**else**

**cout<<"\nNumber found at "<<pos+1<<" position";**

**getch();**

**return 0;**

**}**

**Q7. WAP to implement of a Tree.**

**#include<iostream.h>**

**#include<conio.h>**

**#include<alloc.h>**

**struct bin\_tree**

**{**

**int data;**

**struct bin\_tree \*right,\*left;**

**};**

**typedef struct bin\_tree node;**

**void insert(node \*\*tree,int val)**

**{**

**node \*temp=NULL;**

**if(!(\*tree))**

**{**

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

**temp->left=temp->right=NULL;**

**temp->data=val;**

**\*tree=temp;**

**return;**

**}**

**if(val< (\*tree)->data)**

**{**

**insert(&(\*tree)->left,val);**

**}**

**else**

**if(val>(\*tree)->data){**

**insert(&(\*tree)->right,val);**

**}**

**}**

**void print\_preorder(node \*tree)**

**{**

**if(tree)**

**{**

**cout<<"\n "<<tree->data;**

**print\_preorder(tree->left);**

**print\_preorder(tree->right);**

**}**

**}**

**void print\_inorder(node \*tree)**

**{**

**if(tree)**

**{**

**print\_inorder(tree->left);**

**cout<<"\n "<<tree->data;**

**print\_inorder(tree->right);**

**}**

**}**

**void print\_postorder(node \*tree)**

**{**

**if(tree)**

**{**

**print\_postorder(tree->left);**

**print\_postorder(tree->right);**

**cout<<"\n "<<tree->data;**

**}**

**}**

**void deltree(node \*tree)**

**{**

**if(tree)**

**{**

**deltree(tree->left);**

**deltree(tree->right);**

**free(tree);**

**}**

**}**

**node\* search(node \*\*tree,int val) {**

**if(!(\*tree))**

**{**

**return NULL;**

**}**

**if(val< (\*tree)->data)**

**{**

**search(&((\*tree)->left),val);**

**}**

**else**

**if(val>(\*tree)->data)**

**{**

**search(&((\*tree)->right),val);**

**}**

**else**

**if(val==(\*tree)->data)**

**{**

**return \*tree;**

**}**

**}**

**void main()**

**{**

**clrscr();**

**node \*root;**

**node \*temp;**

**root=NULL;**

**insert(&root,9);**

**insert(&root,4);**

**insert(&root,15);**

**insert(&root,6);**

**insert(&root,12);**

**insert(&root,17);**

**insert(&root,2);**

**cout<<"\n Pre Order Display \n";**

**cout<<"\n Pre Order Display \n";**

**print\_preorder(root);**

**cout<<"\n In Order Display \n";**

**print\_inorder(root);**

**cout<<"\n Post Order Display \n";**

**print\_postorder(root);**

**temp=search(&root,17);**

**if(temp) {**

**cout<<"\n Searched node is: "<<temp->data;**

**}**

**else {**

**cout<<"\n Data not found in tree ";**

**}**

**deltree(root);**

**getch();**

**}**

**Q5. WAP to implement a Doubly Lik-List.**

**#include<iostream.h>**

**#include<stdlib.h>**

**#include<conio.h>**

**#include<alloc.h>**

**struct node**

**{**

**struct node \*prev;**

**struct node \*next;**

**int data;**

**};**

**struct node \*head;**

**void insertion\_beginning();**

**void insertion\_last();**

**void insertion\_specified();**

**void deletion\_beginning();**

**void deletion\_last();**

**void deletion\_specified();**

**void display();**

**void search();**

**void main ()**

**{**

**int choice =0;**

**while(choice != 9)**

**{**

**cout<<"\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n";**

**cout<<"\nChoose one option from the following list ...\n";**

**cout<<"\n1.Insert in begining\n2.Insert at last\n3.Insert at any random location\n4.Delete from Beginning";**

**cout<<"\n5.Delete from last\n6.Delete the node after the given data\n7.Search\n8.Show\n9.Exit\n";**

**cout<<"\nEnter your choice: \n";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**insertion\_beginning();**

**break;**

**case 2:**

**insertion\_last();**

**break;**

**case 3:**

**insertion\_specified();**

**break;**

**case 4:**

**deletion\_beginning();**

**break;**

**case 5:**

**deletion\_last();**

**break;**

**case 6:**

**deletion\_specified();**

**break;**

**case 7:**

**search();**

**break;**

**case 8:**

**display();**

**break;**

**case 9:**

**exit(0);**

**break;**

**default:**

**cout<<"Please enter valid choice..";**

**}**

**}**

**}**

**void insertion\_beginning()**

**{**

**struct node \*ptr;**

**int item;**

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

**if(ptr == NULL)**

**{**

**cout<<"\nOVERFLOW";**

**}**

**else**

**{**

**cout<<"\nEnter Item value";**

**cin>>item;**

**if(head==NULL)**

**{**

**ptr->next = NULL;**

**ptr->prev=NULL;**

**ptr->data=item;**

**head=ptr;**

**}**

**else**

**{**

**ptr->data=item;**

**ptr->prev=NULL;**

**ptr->next = head;**

**head->prev=ptr;**

**head=ptr;**

**}**

**cout<<"\nNode inserted\n";**

**}}**

**void insertion\_last()**

**{**

**struct node \*ptr,\*temp;**

**int item;**

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

**if(ptr == NULL)**

**{**

**cout<<"\nOVERFLOW";**

**}**

**else**

**{**

**cout<<"\nEnter value: ";**

**cin>>item;**

**ptr->data=item;**

**if(head == NULL)**

**{**

**ptr->next = NULL;**

**ptr->prev = NULL;**

**head = ptr;**

**}**

**else**

**{**

**temp = head;**

**while(temp->next!=NULL)**

**{**

**temp = temp->next;**

**}**

**temp->next = ptr;**

**ptr ->prev=temp;**

**ptr->next = NULL;**

**}**

**}**

**cout<<"\nnode inserted\n";**

**}**

**void insertion\_specified()**

**{**

**struct node \*ptr,\*temp;**

**int item,loc,i;**

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

**if(ptr == NULL)**

**{**

**cout<<"\n OVERFLOW";**

**}**

**else**

**{**

**temp=head;**

**cout<<"Enter the location: ";**

**cin>>loc;**

**for(i=0;i<loc;i++)**

**{**

**temp = temp->next;**

**if(temp == NULL)**

**{**

**cout<<"\n There are less elements";**

**return;**

**}**

**}**

**cout<<"Enter value: ";**

**cin>>item;**

**ptr->data = item;**

**ptr->next = temp->next;**

**ptr -> prev = temp;**

**temp->next = ptr;**

**temp->next->prev=ptr;**

**cout<<"\nnode inserted\n";**

**}**

**}**

**void deletion\_beginning()**

**{**

**struct node \*ptr;**

**if(head == NULL)**

**{**

**cout<<"\n UNDERFLOW";**

**}**

**else if(head->next == NULL)**

**{**

**head = NULL;**

**free(head);**

**cout<<"\nnode deleted\n";**

**}**

**else**

**{**

**ptr = head;**

**head = head -> next;**

**head -> prev = NULL;**

**free(ptr);**

**cout<<"\nnode deleted\n";**

**}**

**}**

**void deletion\_last()**

**{**

**struct node \*ptr;**

**if(head == NULL)**

**{**

**cout<<"\n UNDERFLOW";**

**}**

**else if(head->next == NULL)**

**{**

**head = NULL;**

**free(head);**

**cout<<"\nnode deleted\n";**

**}**

**else**

**{**

**ptr = head;**

**if(ptr->next != NULL)**

**{**

**ptr = ptr -> next;**

**}**

**ptr -> prev -> next = NULL;**

**free(ptr);**

**cout<<"\nnode deleted\n";**

**}**

**}**

**void deletion\_specified()**

**{**

**struct node \*ptr, \*temp;**

**int val;**

**cout<<"\n Enter the data after which the node is to be deleted : ";**

**cin>>val;**

**ptr = head;**

**while(ptr -> data != val)**

**ptr = ptr -> next;**

**if(ptr -> next == NULL)**

**{**

**cout<<"\nCan't delete\n";**

**}**

**else if(ptr -> next -> next == NULL)**

**{**

**ptr ->next = NULL;**

**}**

**else**

**{**

**temp = ptr -> next;**

**ptr -> next = temp -> next;**

**temp -> next -> prev = ptr;**

**free(temp);**

**cout<<"\nnode deleted\n";**

**}**

**}**

**void display()**

**{**

**struct node \*ptr;**

**cout<<"\n printing values...\n";**

**ptr = head;**

**while(ptr != NULL)**

**{**

**cout<<ptr->data;**

**ptr=ptr->next;**

**}**

**}**

**void search()**

**{**

**struct node \*ptr;**

**int item,i=0,flag;**

**ptr = head;**

**if(ptr == NULL)**

**{**

**cout<<"\nEmpty List\n";**

**}**

**else**

**{**

**cout<<"\nEnter item which you want to search: ";**

**cin>>item;**

**while (ptr!=NULL)**

**{**

**if(ptr->data == item)**

**{**

**cout<<"\nitem found ";**

**flag=0;**

**break;**

**}**

**else**

**{**

**flag=1;**

**} i++;**

**ptr = ptr -> next;**

**}**

**if(flag==1)**

**{**

**cout<<"\nItem not found\n";**

**}**

**}**

**}**

**Q6. WAP to implement a Circular queue using Array.**

**#include<iostream.h>**

**#include<iostream.h>**

**#include<conio.h>**

**#include<alloc.h>**

**#include<process.h>**

**struct node**

**{**

**int data;**

**struct node \*next;**

**};**

**struct node \*head;**

**void beg\_insert();**

**void last\_insert();**

**void begin\_delete();**

**void last\_delete();**

**void display();**

**void search();**

**void main()**

**{**

**int choice=0;**

**clrscr();**

**cout<<"\n \*\*\*\*\*\*\*\*\*\*\* Main Menu \*\*\*\*\*\*\*\*\*\*\*\*\* \n";**

**cout<<"\n1. Insert in Beginning \n";**

**cout<<"\n2. Insert at last ";**

**cout<<"\n3. Delete from beginning ";**

**cout<<"\n4. Delete from last ";**

**cout<<"\n5. Search for an element ";**

**cout<<"\n6. Show ";**

**cout<<"\n7. Exit ";**

**do**

**{**

**cout<<"\n Enter your choice: ";**

**cin>>choice;**

**switch(choice)**

**{**

**case 1:**

**beg\_insert();**

**break;**

**case 2:**

**last\_insert();**

**break;**

**case 3:**

**begin\_delete();**

**break;**

**case 4:**

**last\_delete();**

**break;**

**case 5:**

**search();**

**break;**

**case 6:**

**display();**

**break;**

**case 7:**

**exit(0);**

**break;**

**default:**

**cout<<"\n Wrong Choice ";**

**}**

**}while(choice!=7);**

**getch();**

**}**

**void beg\_insert()**

**{**

**struct node \*ptr,\*temp;**

**int item;**

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

**if(ptr==NULL)**

**{**

**cout<<"\n Overflow";**

**}**

**else**

**{**

**cout<<"\n Enter the node data: ";**

**cin>>item;**

**ptr->data=item;**

**if(head==NULL)**

**{**

**head=ptr;**

**ptr->next=head;**

**}**

**else**

**{**

**temp=head;**

**while(temp->next!=head)**

**temp=temp->next;**

**ptr->next=head;**

**temp->next=ptr;**

**head=ptr;**

**}**

**cout<<"\n Node inserted";**

**}**

**}**

**void last\_insert()**

**{**

**struct node \*ptr,\*temp;**

**int item;**

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

**if(ptr==NULL)**

**{**

**cout<<"\n Overflow ";**

**}**

**else**

**{**

**cout<<"\n Enter Data: ";**

**cin>>item;**

**ptr->data=item;**

**if(head==NULL)**

**{**

**head=ptr;**

**ptr->next=head;**

**}**

**else**

**{**

**temp=head;**

**while(temp->next!=head)**

**{**

**temp=temp->next;**

**}**

**temp->next=ptr;**

**ptr->next=head;**

**}**

**cout<<"\n Node inserted ";**

**}**

**}**

**void begin\_delete()**

**{**

**struct node \*ptr;**

**if(head==NULL)**

**{**

**cout<<"\n Underfow ";**

**}**

**else**

**if(head->next==head)**

**{**

**head=NULL;**

**free(head);**

**cout<<"\n Node deleted ";**

**}**

**else**

**{**

**ptr=head;**

**while(ptr->next!=head)**

**ptr=ptr->next;**

**ptr->next=head->next;**

**free(head);**

**head=ptr->next;**

**cout<<"\n Node deleted";**

**}**

**}**

**void last\_delete()**

**{**

**struct node \*ptr,\*preptr;**

**if(head==NULL)**

**{**

**cout<<"\n Underflow ";**

**}**

**else**

**if(head->next==head)**

**{**

**head=NULL;**

**free(head);**

**cout<<"\n Node deleted";**

**}**

**else**

**{**

**ptr=head;**

**while(ptr->next!=head)**

**{**

**preptr=ptr;**

**ptr=ptr->next;**

**}**

**preptr->next=ptr->next;**

**free(ptr);**

**cout<<"\n Node deleted";**

**}**

**}**

**void search()**

**{**

**struct node \*ptr;**

**int item,i=0,flag=1;**

**ptr=head;**

**if(ptr==NULL)**

**{**

**cout<<"\n Empty List ";**

**}**

**else**

**{**

**cout<<"\n Enter item which you want to search: ";**

**cin>>item;**

**if(head->data==item)**

**{**

**cout<<"\n Item found at location: "<<i+1;**

**flag=0;**

**}**

**else**

**{**

**while(ptr->next!=head)**

**{**

**if(ptr->data==item)**

**{**

**cout<<"\n Item found at location: "<<i+1;**

**flag=0;**

**break;**

**}**

**else {**

**flag=1;**

**}**

**i++;**

**ptr=ptr->next;**

**}**

**}**

**if(flag!=0)**

**{**

**cout<<"\n Item not found";**

**}**

**}**

**}**

**void display()**

**{**

**struct node \*ptr;**

**ptr=head;**

**if(head==NULL)**

**{**

**cout<<"\n Nothing to print ";**

**}**

**else**

**{**

**cout<<"\n Printing Values: ";**

**while(ptr->next!=head)**

**{**

**cout<<"\n"<<ptr->data;**

**ptr=ptr->next;**

**}**

**cout<<"\n"<<ptr->data;**

**}**

**}**