**Assignment-1**

1. Write a program for Binary Search.

#include <stdio.h>

#include<stdlib.h>

int main()

{

int n;

scanf("%d",&n);

int a[n],target;

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

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

}

scanf("%d",&target);

int first=0,last=n-1,mid;

while(first<=last){

mid=(last+first)/2;

if(a[mid]==target){

printf("Found at index: %d",mid);

exit(0);

}

else if(a[mid]>target){

last=mid-1;

}

else{

first=mid+1;

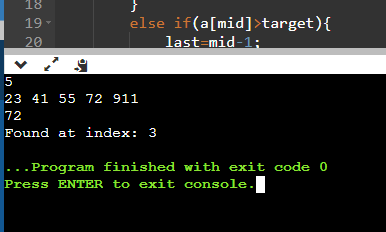
}

}

printf("Element not found");

return 0;

}



1. Write a program for Binary search using recursion.

#include <stdio.h>

#include<stdlib.h>

int Binary\_search(int arr[],int target,int first,int last){

int mid=(first+last)/2;

if(first>last){

return -1;

}

else if(arr[mid]==target){

return mid;

}

else if(arr[mid]<target){

return Binary\_search(arr,target,mid+1,last);

}

return Binary\_search(arr,target,first,mid-1);

}

int main()

{

int n;

scanf("%d",&n);

int a[n],target;

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

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

}

scanf("%d",&target);

int c=Binary\_search(a,target,0,n-1);

if(c==-1){

printf("Element not found");

}

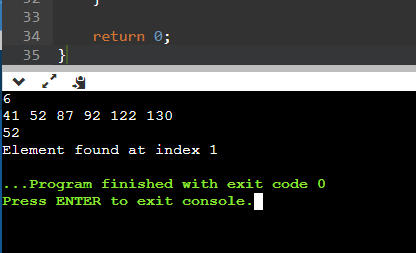
else{

printf("Element found at index %d",c);

}

return 0;

}



1. Write a program for bubble sort and display all the elements in each iteration,and also compute the complexity analysis.

#include <stdio.h>

#include<stdlib.h>

int main()

{

int n;

scanf("%d",&n);

int a[n],temp;

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

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

}

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

for(int j=i+1;j<n;j++){

if(a[i]>a[j]){

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

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

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

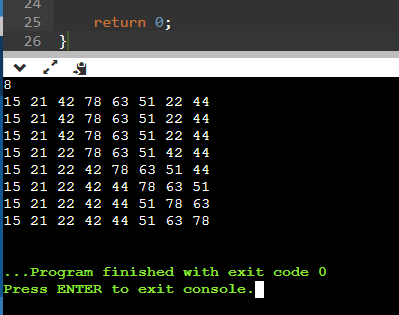
}

printf("\n");

}

return 0;

}



Best Case: When the elements are in sorted ordered, then we can check in one iteration whether the array needs sorting or not and if not, we can terminate the loop. O(N)

Average and worst case: In this case, elements are either reversely sorted or not sorted. First loop will iterate for N-1 times and second loop will iterate for N times.

N\*N-1=N^2 -N => O(N^2).

1. Write a program for Selection sort and display all the elements in each iteration,and also compute the complexity analysis.

#include <stdio.h>

#include<stdlib.h>

int main()

{

int n,min;

scanf("%d",&n);

int a[n],temp;

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

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

}

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

min=i;

for(int j=i+1;j<n;j++){

if(a[min]>a[j]){

min=j;

}

}

if(min!=i){

temp=a[min];

a[min]=a[i];

a[i]=temp;

}

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

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

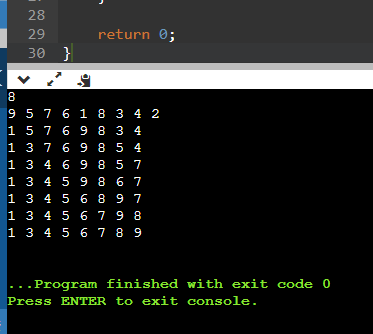
}

printf("\n");

}

return 0;

}



Complexity analysis:

Best case: To check whether an array is sorted or not, one loop is enough. So when the array is sorted, the time complexity is O(N).

Average and worst case: There are two for loops and each iterate for N times. Therefore, it is O(N^2) time complexity.

1. Given an integer array, write an algorithm to find max and second max. Calculate its step Count. Express the step count Big O,Theeta and Omega.

#include <stdio.h>

#include<limits.h>

int main()

{

int n,max1=INT\_MIN,max2=INT\_MIN;

scanf("%d",&n);

int a[n],temp;

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

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

}

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

if(a[i]>max1){

max2=max1;

max1=a[i];

}

else if(a[i]>max2){

max2=a[i];

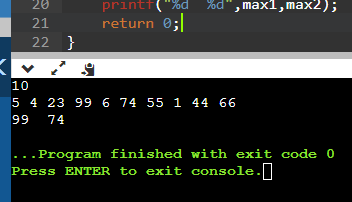
}

}

printf("%d %d",max1,max2);

return 0;

}



Complexity Analysis:

There is only single loop with proper increment. Therefore, the complexity is O(N).

1. For a given singly linked list, insert a node a. At the beginning b. At the end c. At a position k.

#include <stdio.h>

#include<stdlib.h>

struct Node{

int val;

struct Node \*next;

};

struct Node\* InsertAtBegin(struct Node\* head,int info){

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

temp->val=info;

if(head==NULL){

temp->next=NULL;

}

else{

temp->next=head;

}

head=temp;

return head;

}

struct Node\* InsertAtEnd(struct Node\* head,int info){

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

temp->val=info;

temp->next=NULL;

struct Node \*ptr=head;

if(head==NULL){

head=temp;

}

else{

while(ptr->next){

ptr=ptr->next;

}

ptr->next=temp;

}

return head;

}

struct Node\* InsertAtK(struct Node\* head,int info,int k){

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

temp->val=info;

temp->next=NULL;

int c=1;

struct Node \*ptr=head;

if(k==1){

head=temp;

}

else{

while(c<k-1){

ptr=ptr->next;

}

temp->next=ptr->next;

ptr->next=temp;

}

return head;

}

int main()

{

struct Node\* head=NULL;

head=InsertAtBegin(head,10);

head=InsertAtBegin(head,20);

head=InsertAtEnd(head,30);

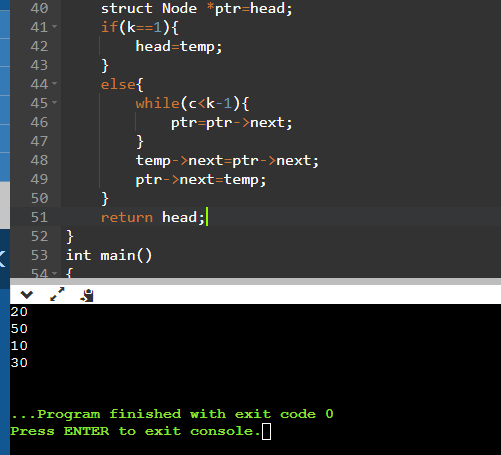
head=InsertAtK(head,50,2);

for(struct Node \*ptr=head;ptr!=NULL;ptr=ptr->next){

printf("%d\n",ptr->val);

}

}



1. For a given linked list delete a node at a.beginning b. at the end c. at the position k

#include <stdio.h>

#include<stdlib.h>

struct Node{

int val;

struct Node \*next;

};

struct Node\* head=NULL;

void InsertAtBegin(int info){

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

temp->val=info;

if(head==NULL){

temp->next=NULL;

}

else{

temp->next=head;

}

head=temp;

}

void deleteAtBegin(){

if(head==NULL){

printf("Empty list");

}

else{

struct Node\* temp=head;

head=head->next;

free(temp);

}

}

void deleteAtEnd(){

if(head==NULL){

printf("Empty");

}

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

head=NULL;

}

else{

struct Node\* ptr=head;

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

ptr=ptr->next;

}

ptr->next=NULL;

}

}

void deleteAtK(int k){

struct Node \*ptr=head,\*q=NULL;

if(head==NULL){

printf("Empty list");

}

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

head=NULL;

}

else{

while(k--){

q=ptr;

ptr=ptr->next;

if(ptr->next==NULL){

}

}

q->next=ptr->next;

free(ptr);

}

}

int main()

{

InsertAtBegin(10);

InsertAtBegin(20);

InsertAtBegin(30);

InsertAtBegin(40);

InsertAtBegin(50);

deleteAtBegin();

deleteAtK(1);

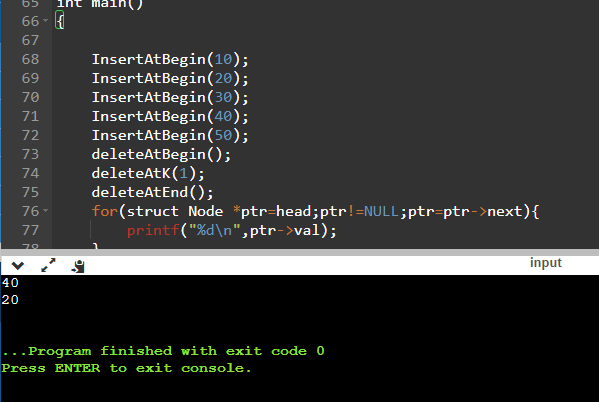
deleteAtEnd();

for(struct Node \*ptr=head;ptr!=NULL;ptr=ptr->next){

printf("%d\n",ptr->val);

}

}



8)Reverse a Singly Linked List

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* link;

};

struct Node\* root;

int len,max,min;

void Insert(void);

void Display(void);

void ReverseList(void);

int main()

{

int ch;

printf("Single Linked list Operation\n");

printf("1.Insert\n");

printf("2.Display\n");

printf("3.ReverseList\n");

printf("4.Quit\n");

while(1){

printf("Enter your Choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: Insert();

break;

case 2: Display();

break;

case 3: ReverseList();

break;

case 4: exit(1);

break;

default: printf("Invalid input\n\n");

}

}

return 0;

}

int length(){

int count=0;

struct Node\* temp;

temp= root;

while(temp!=NULL)

{

count++;

temp=temp->link;

}

return count;

}

void Insert(){

struct Node\* temp;

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

printf("Enter the number");

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

temp->link=NULL;

if(root==NULL){

root=temp;

}

else{

struct Node\* p;

p=root;

while(p->link!=NULL){

p=p->link;

}

p->link=temp;

}

}

void Display(){

struct Node\* temp;

temp=root;

if(temp== NULL){

printf("List is Empty");

}

else{

while(temp != NULL)

{

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

temp = temp->link;

}

printf("\n");

}

}

void ReverseList(){

int i,j,len,temp,k;

struct Node \*p, \*q;

len=length();

i=0;

j=len-1;

p=q=root;

while(i<j){

k=0;

while(k<j){

q=q->link;

k++;

}

temp = p->data;

p->data = q->data;

q->data= temp;

i++;

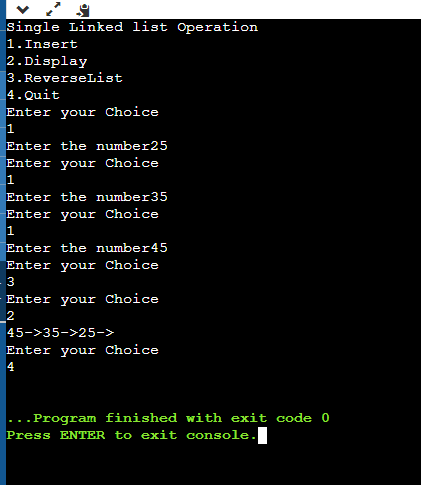
j--;

p= p->link;

q=root;

}

}



1. Write a program for printing the following in a given linked list c. Maximum-Minimum

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* link;

};

struct Node\* root;

int len,max,min;

void append(void);

void Display(void);

int maxminoflist(void);

int main()

{

int ch;

printf("Single Linked list Operation\n");

printf("1.Append\n");

printf("2.Display\n");

printf("3.maximinoflist\n");

printf("4.Quit\n");

while(1){

printf("Enter your Choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: append();

break;

case 2: Display();

break;

case 3: max,min=maxminoflist();

printf("print the maximum of list %d\nprint the minimum of list %d\n\n",max,min);

break;

case 4: exit(1);

break;

default: printf("Invalid input\n\n");

}

}

return 0;

}

void append(){

struct Node\* temp;

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

printf("Enter the number");

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

temp->link=NULL;

if(root==NULL){

root=temp;

}

else{

struct Node\* p;

p=root;

while(p->link!=NULL){

p=p->link;

}

p->link=temp;

}

}

void Display(){

struct Node\* temp;

temp=root;

if(temp== NULL){

printf("List is Empty");

}

else{

while(temp != NULL)

{

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

temp = temp->link;

}

printf("\n");

}

}

int maxminoflist()

{

int i;

struct Node\*p;

i=1;

p=root;

while(p!=NULL){

if(i==1){

max=p->data;

min=p->data;

i=0;

}

else{

if(max<p->data){

max=p->data;

}

if(min>p->data){

min=p->data;

}

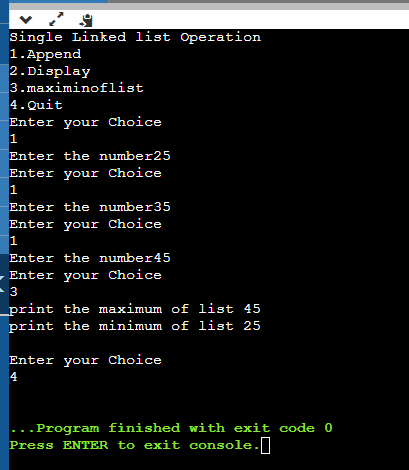
}

p=p->link;

}

return max,min;

}



10)Take N numbers as input from the user and create a doubly linked list

#include <stdio.h>

#include <stdlib.h>

struct Node{

int data;

struct Node\*right;

struct Node\*left;

};

struct Node\*root=NULL;

int N,i,len;

int min;

void Append(void);

int length(void);

void Display(void);

int main()

{

int ch;

printf("Double linked list\n");

printf("1.Append\n");

printf("2.length\n");

printf("3.Display\n");

printf("4.Quit\n");

while(1){

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch){

case 1: printf("Enter the value of N\n");

scanf("%d",&N);

for(i=0;i<N;i++){

Append();

}

break;

case 2: len=length();

printf("length of the linked list %d\n",len);

break;

case 3: Display();

break;

case 4: exit(1);

break;

default: printf("Invalid input\n\n");

}

}

}

void Append(){

struct Node\*temp;

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

printf("Enter the number\n");

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

temp->right=NULL;

temp->left=NULL;

if(root==NULL){

root=temp;

}

else

{

struct Node\*p;

p=root;

while(p->right!=NULL)

{

p=p->right;

}

p->right = temp;

temp->left = p;

}

}

int length(){

struct Node\* temp;

int count=0;

temp=root;

while(temp!=NULL){

count++;

temp=temp->right;

}

return count;

}

void Display(){

struct Node\* temp;

temp=root;

if(temp== NULL){

printf("List is Empty");

}

else{

while(temp != NULL)

{

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

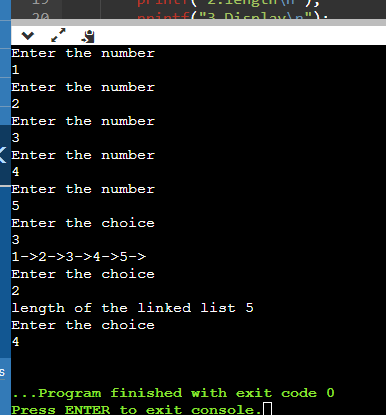
temp = temp->right;

}

printf("\n");

}

}



11)Find the smallest number in the doubly linked list.

#include <stdio.h>

#include <stdlib.h>

struct Node{

int data;

struct Node\*right;

struct Node\*left;

};

struct Node\*root=NULL;

int N,i,len;

int min;

void Append(void);

int length(void);

void Display(void);

int minimumoflist(void);

int main()

{

int ch;

printf("Double linked list\n");

printf("1.Append\n");

printf("2.length\n");

printf("3.Display\n");

printf("4.minimumoflist\n");

printf("5.Quit\n");

while(1){

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch){

case 1: printf("Enter the value of N\n");

scanf("%d",&N);

for(i=0;i<N;i++){

Append();

}

break;

case 2: len=length();

printf("length of the linked list %d\n",len);

break;

case 3: Display();

break;

case 4: min=minimumoflist();

printf("print minimum %d\n",min);

break;

case 5: exit(1);

break;

default: printf("Invalid input\n\n");

}

}

}

void Append(){

struct Node\*temp;

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

printf("Enter the number\n");

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

temp->right=NULL;

temp->left=NULL;

if(root==NULL){

root=temp;

}

else

{

struct Node\*p;

p=root;

while(p->right!=NULL)

{

p=p->right;

}

p->right = temp;

temp->left = p;

}

}

int length(){

struct Node\* temp;

int count=0;

temp=root;

while(temp!=NULL){

count++;

temp=temp->right;

}

return count;

}

void Display(){

struct Node\* temp;

temp=root;

if(temp== NULL){

printf("List is Empty");

}

else{

while(temp != NULL)

{

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

temp = temp->right;

}

printf("\n");

}

}

int minimumoflist()

{

int i;

struct Node\*p;

i=1;

p=root;

while(p!=NULL)

{

if(i==1){

min=p->data;

i=0;

}

else{

if(min>p->data)

{

min=p->data;

}

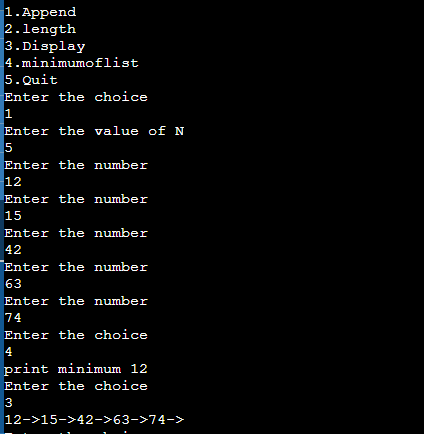
}

p=p->right;

}

return min;

}



12) Reverse a doubly linked list

#include <stdio.h>

#include <stdlib.h>

struct Node{

int data;

struct Node\*right;

struct Node\*left;

};

struct Node\*root=NULL;

int N,i,len;

int max,min;

void Append(void);

int length(void);

void Display(void);

void reverselist(void);

int main()

{

int ch;

printf("Double linked list\n");

printf("1.Append\n");

printf("2.length\n");

printf("3.Display\n");

printf("4.reverselist\n");

printf("5.Quit\n");

while(1){

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch){

case 1: printf("Enter the value of N\n");

scanf("%d",&N);

for(i=0;i<N;i++){

Append();

}

break;

case 2: len=length();

printf("length of the linked list %d\n",len);

break;

case 3: Display();

break;

case 4: reverselist();

break;

case 5: exit(1);

break;

default: printf("Invalid input\n\n");

}

}

}

void Append(){

struct Node\*temp;

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

printf("Enter the number\n");

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

temp->right=NULL;

temp->left=NULL;

if(root==NULL){

root=temp;

}

else

{

struct Node\*p;

p=root;

while(p->right!=NULL)

{

p=p->right;

}

p->right = temp;

temp->left = p;

}

}

int length(){

struct Node\* temp;

int count=0;

temp=root;

while(temp!=NULL){

count++;

temp=temp->right;

}

return count;

}

void Display(){

struct Node\* temp;

temp=root;

if(temp== NULL){

printf("List is Empty");

}

else{

while(temp != NULL)

{

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

temp = temp->right;

}

printf("\n");

}

}

void reverselist(){

struct Node \*p,\*q,\*tail;

p=root;

tail=root;

while(tail->right!=NULL){

tail=tail->right;

}

while(p!=NULL){

q=p->right;

p->right=p->left;

p->left=q;

p=q;

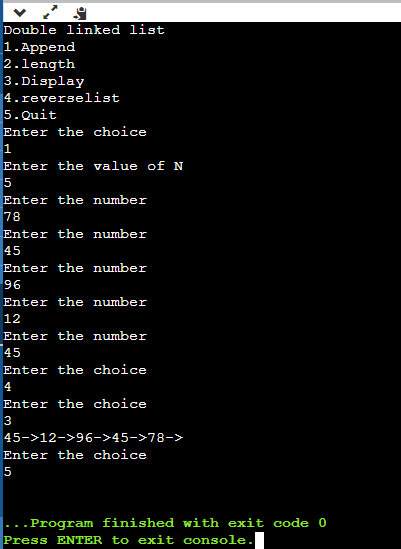
}

p=root;

root=tail;

tail=p;

}



1. Implement stack using array and linked list.

#include <stdio.h>

#include<stdlib.h>

int stack[100], n=100, top=-1;

void push(int val) {

if(top>=n-1)

printf("Stack overflow");

else {

top++;

stack[top]=val;

}

}

void pop() {

if(top<=-1)

printf("Stack underflow\n");

else {

printf("The popped element is %d\n",stack[top]);

top--;

}

}

void display() {

if(top>=0) {

printf("Stack elements are:\n");

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

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

} else

printf("Empty stack");

}

int main() {

int ch, val;

printf("1) Push\n");

printf("2) Pop\n");

printf("3) Display\n");

printf("4) Exit\n");

do {

printf("Enter choice: ");

scanf("%d",&ch);

switch(ch) {

case 1: {

printf("Enter value to be pushed:");

scanf("%d",&val);

push(val);

break;

}

case 2: {

pop();

break;

}

case 3: {

display();

break;

}

case 4: {

printf("Exit");

break;

}

default: {

printf("Invalid Choice");

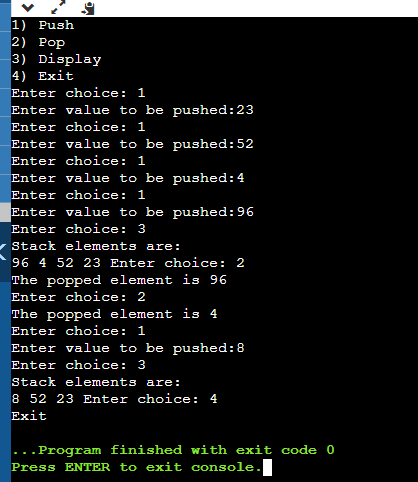
}

}

}while(ch!=4);

return 0;

}



//stack implementation using linked list.

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* link;

};

struct Node\* top=NULL;

void push();

void pop();

void traverse();

void main()

{

while(1)

{

int ch;

printf("1.push\n");

printf("2.pop\n");

printf("3.traverse\n");

printf("4.Quit\n");

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: push();

break;

case 2: pop();

break;

case 3: traverse();

break;

case 4: exit(1);

break;

default: printf("Invalid Input\n");

}

}

}

void push()

{

struct Node\*temp;

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

printf("Enter node item:");

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

temp->link=top;

top=temp;

}

void pop()

{

struct Node\*temp;

if(top==NULL){

printf("Stack is empty\n");

}

else

{

temp=top;

printf("popped %d\n",temp->data);

top=top->link; //temp->link

temp->link=NULL;

free(temp);

}

}

void traverse()

{

struct Node\*temp;

if(top==NULL)

{

printf("Stack is empty\n");

}

else

{

temp=top;

printf("The stack elements\n");

while(temp!=NULL)

{

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

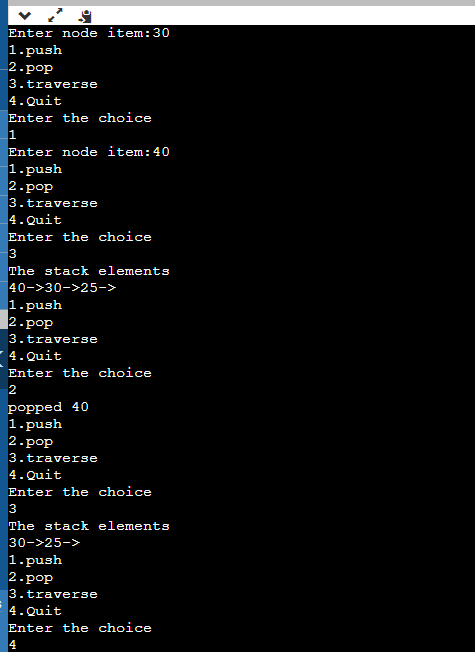
temp=temp->link;

}

printf("\n");

}

}



1. Implement queue using array and linked list.

#include <stdio.h>

#include<stdlib.h>

int queue[100], n = 100, front = - 1, rear = - 1;

void Insert() {

int val;

if (rear == n - 1)

printf("Queue Overflow\n");

else {

if (front == - 1)

front = 0;

printf("Insert the element in queue : ");

scanf("%d",&val);

rear++;

queue[rear] = val;

}

}

void Delete() {

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

printf("Queue Underflow \n");

return ;

} else {

printf("Element deleted from queue is :%d\n ",queue[front]);

front++;;

}

}

void Display() {

if (front == - 1)

printf("Queue is empty\n");

else {

printf("Queue elements are : ");

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

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

printf("\n");

}

}

int main() {

int ch;

printf("1.Insert\n");

printf("2.Delete\n");

printf("3.Display\n");

printf("4.Exit\n");

do {

printf("Enter choice:\n");

scanf("%d",&ch);

switch (ch) {

case 1: Insert();

break;

case 2: Delete();

break;

case 3: Display();

break;

case 4:printf("Exit\n");

break;

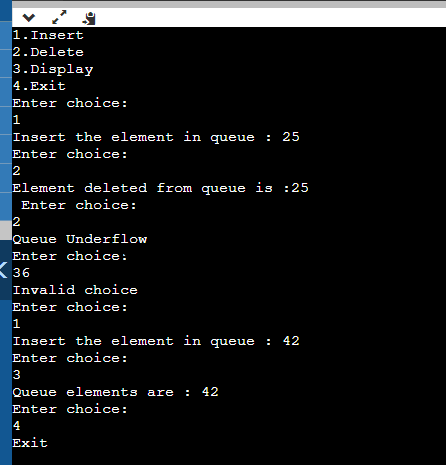
default: printf("Invalid choice\n");

}

} while(ch!=4);

return 0;

}



//queue using linked list

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* link;

};

struct Node \*rear=NULL;

struct Node \*front=NULL;

void insert();

void delete();

void traverse();

void main()

{

printf("1.insert\n");

printf("2.delete\n");

printf("3.traverse\n");

printf("4.Quit\n");

while(1)

{

int ch;

printf("Enter the choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1: insert();

break;

case 2: delete();

break;

case 3: traverse();

break;

case 4: exit(1);

break;

default: printf("Invalid Input\n");

}

}

}

void insert()

{

struct Node\*temp;

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

printf("Enter node data\n");

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

temp->link=NULL;

if(front==0 && rear==0)

{

front=rear=temp;

}

else

{

rear->link=temp;

rear=temp;

}

}

void delete()

{

struct Node\*temp;

temp=front;

if(front==0 && rear==0)

{

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

}

else

{

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

front=front->link;

free(temp);

}

printf("\n");

}

void traverse()

{

struct Node\*temp;

if(front==0 && rear==0)

{

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

}

else

{

printf("The Queue Elements\n");

temp=front;

while(temp!=NULL)

{

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

temp=temp->link;

}

printf("\n");

}

}

