Day 1

1. Write a C program to find the given number is odd or even.

#include<stdio.h>

int main()

{

int n;

printf("enter the number");

scanf("%d",&n);

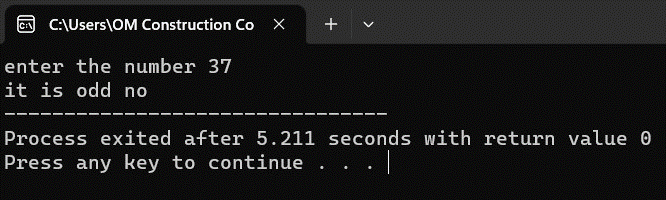
if(n%2==0)

printf("it is even no");

else

printf("it is odd no");

}



1. Write a C program to find sum of natural number using foor loop.

#include<stdio.h>

int main()

{

int i,n,sum;

printf("enert the number");

scanf("%d",&n);

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

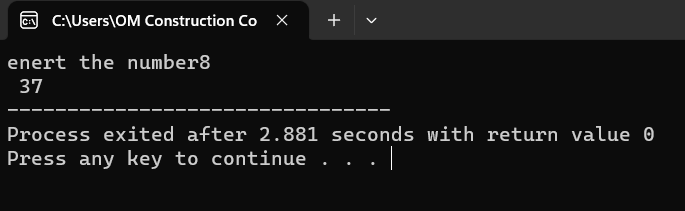
{

sum=sum+i;

}

printf(" %d",sum);

}



1. Write a C program to find sum of even natural number using while loop.

#include<stdio.h>

int main()

{

int i=2,n,sum=0;

printf("enter the number");

scanf("%d",&n);

while(i<=n)

{

if(i%2==0)

{

sum+=i;

}

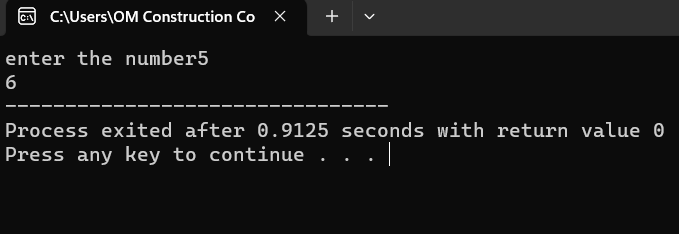
++i;

}

printf("%d",sum);

return 0;

}



1. Write a C program to reverse the given number.

#include<stdio.h>

int main()

{

int n,rev=0,rem;

printf("enter the number");

scanf("%d",&n);

while(n!=0)

{

rem=n%10;

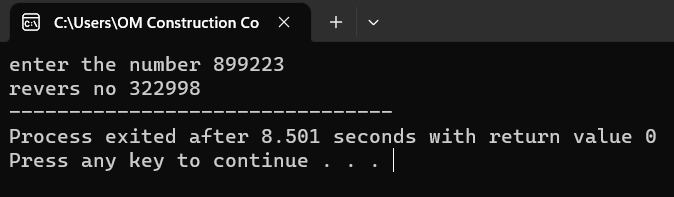
rev=rev\*10+rem;

n=n/10;

}

printf("revers no %d",rev);

}



1. Write a C program to check the given number is palindrome number or not.

#include<stdio.h>

int main()

{

int orginaln,n,rem,sum=0;

printf("enter the number");

scanf("%d",&n);

orginaln=n;

while(n!=0)

{

rem=n%10;

sum=sum\*10+rem;

n=n/10;

}

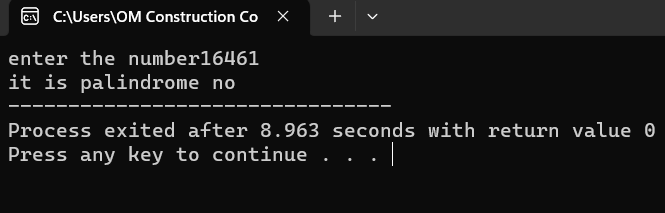
if(orginaln==sum)

printf("it is palindrome no");

else

printf("it is not an palindrome no ");

}



1. Write a C program to check the given number is armstrong number or not.

#include<stdio.h>

int main()

{

int temp,n,rem,sum=0;

printf("enter the number");

scanf("%d",&n);

temp=n;

while(n>0)

{

rem=n%10;

sum=sum+(rem\*rem\*rem);

n=n/10;

}

if(temp==sum)

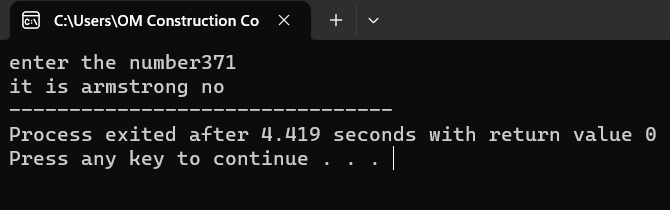
printf("it is armstrong no ");

else

printf("it is not an armstrong no");

return 0;

}



1. Write a C program to find factorial of given number with recursion.

#include<stdio.h>

#include<conio.h>

int fact(int number)

{

if(number==0 || number==1)

return 1;

else

return(number \* fact(number-1));

}

int main()

{

int n,f;

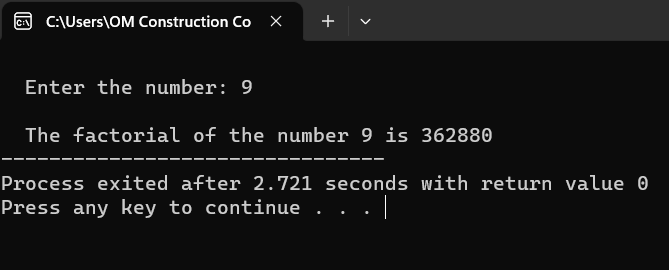
printf("\n Enter the number: ");

scanf("%d",&n);

f=fact(n);

printf("\n The factorial of the number %d is %d",n,f);

}



1. Write a C program to find factorial of given number without recursion.

#include<stdio.h>

int main() {

int n, i;

long factorial = 1;

printf("Enter an no ");

scanf("%d",&n);

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

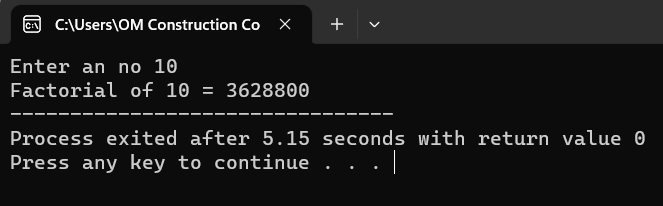
factorial \*= i;

}

printf("Factorial of %d = %llu", n, factorial);

return 0;

}



1. Write a C program to generate Fibonacci series with recursion.

#include <stdio.h>

int Fibonacci(int n)

{

static int n1= 0,n2=1,n3;

if(n > 0){

n3 = n1 + n2;

n1 = n2;

n2 = n3;

printf("%d",n3);

Fibonacci(n-1);

}

}

int main(){

int n;

printf("Enter the number of elements: ");

scanf("%d", &n);

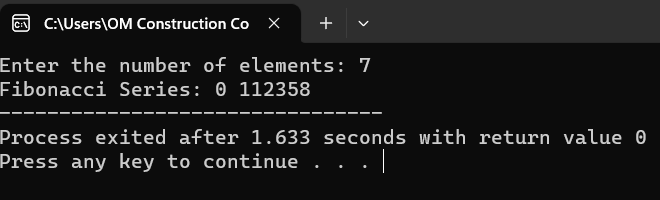
printf("Fibonacci Series: ");

printf("%d %d",0,1);

Fibonacci(n-2);

return 0;

}



1. Write a C program to generate Fibonacci series without recursion.

#include<stdio.h>

int main()

{

int a=0,b=1,result=0,n,i;

printf("enter the number");

scanf("%d",&n);

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

{

printf("%d",a);

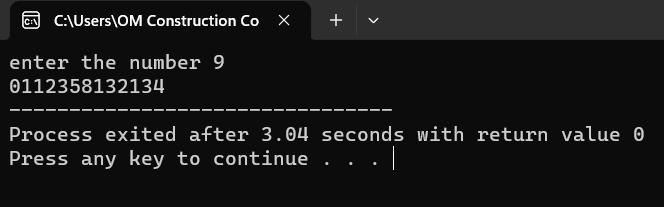
result=a+b;

a=b;

b=result;

}

}



1. Write a C program to search the particular element array using Linear search.

#include <stdio.h>

int linearSearch(int array[], int size, int element) {

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

if(array[i] == element)

return i;

}

return -1;

}

int main() {

int array[] = {1, 3, 5, 7, 9}, element;

int size = sizeof(array) / sizeof(array[0]);

printf("Enter the element to be searched: ");

scanf("%d", &element);

int searchResult = linearSearch(array, size, element);

if(searchResult != -1)

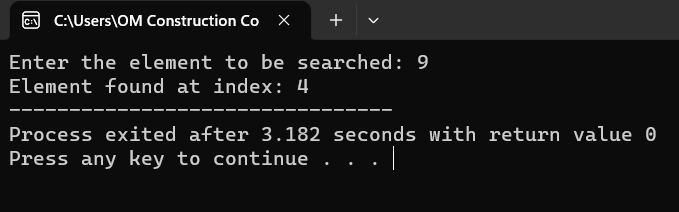
printf("Element found at index: %d", searchResult);

else

printf("Element not found in the array.");

return 0;

}



1. Write a C program to search the particular element array using Binary search.

#include <stdio.h>

int start = 0, end = size-1;

while(start <= end) {

int mid = (start + end) / 2;

if (array[mid] == element)

return mid;

if (array[mid] < element)

start = mid + 1;

else

end = mid - 1;

}

return -1;

}

int main() {

int array[] = {387,333,331,987,689}, element;

int size = sizeof(array) / sizeof(array[0]);

printf("Enter the element to be searched: ");

scanf("%d", &element);

int searchResult = binarySearch(array, size, element);

if(searchResult != -1)

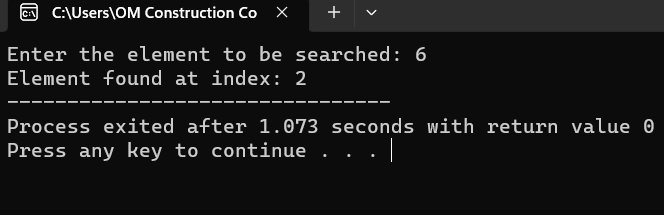
printf("Element found at index: %d", searchResult);

else

printf("Element not found in the array.");

return 0;

}



Day 2

1. Write a C program to count number of times A E I O U present in the given string.

#include <stdio.h>

#include <string.h>

int main()

{

char str[100];

int vowels = 0;

printf("Enter a string: ");

fgets(str, 100, stdin);

for(int i = 0; i < strlen(str); i++)

{

if(str[i] == 'a' || str[i] == 'e' || str[i] == 'i' || str[i] == 'o' || str[i] == 'u' || str[i] == 'A' || str[i] == 'E' || str[i] == 'I' || str[i] == 'O' || str[i] == 'U')

{

vowels++;

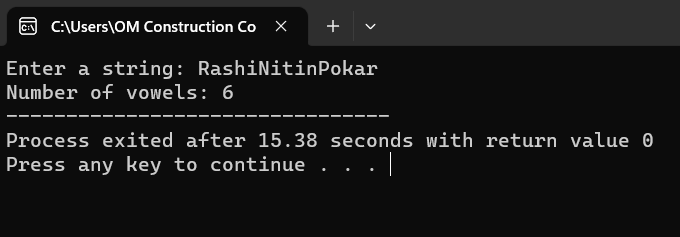
}

}

printf("Number of vowels: %d", vowels);

return 0;

}



1. Write a C program to calculate sum of elements in an array.

#include<stdio.h>

#include<string.h>

int main()

{

int a[100],i,n,sum=0;

printf("Enter size of the array : ");

scanf("%d",&n);

printf("Enter elements in array : ");

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

{

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

}

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

{

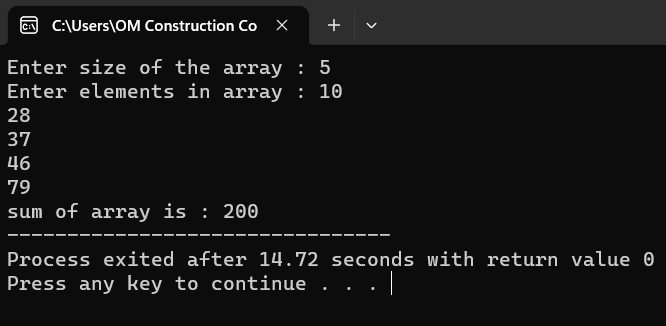
sum+=a[i];

}

printf("sum of array is : %d",sum);

return 0;

}



1. Write a C program to merge 2 arrays.

#include<stdio.h>

#include<conio.h>

int main()

{

int arr1[50], arr2[50], size1, size2, i, k, merge[100];

printf("Enter Array 1 Size: ");

scanf("%d", &size1);

printf("Enter Array 1 Elements: ");

for(i=0; i<size1; i++)

{

scanf("%d", &arr1[i]);

merge[i] = arr1[i];

}

k = i;

printf("\nEnter Array 2 Size: ");

scanf("%d", &size2);

printf("Enter Array 2 Elements: ");

for(i=0; i<size2; i++)

{

scanf("%d", &arr2[i]);

merge[k] = arr2[i];

k++;

}

printf("\nThe new array after merging is:\n");

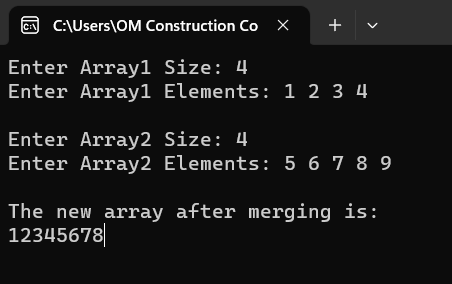
for(i=0; i<k; i++)

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

getch();

return 0;

}



1. Write a C program to reverse a string.

#include<stdio.h>

#include<string.h>

int main ()

{

char str[100];

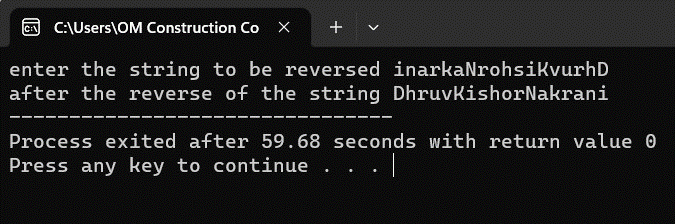
printf("enter the string to be reversed ");

scanf("%s",&str);

printf("after the reverse of the string %s",strrev(str));

return 0;

}



1. Write a C program to insert an element in middle of array.

#include <stdio.h>

#include <conio.h>

int main ()

{

int arr[50];

int pos, i, num;

printf (" \n Enter the number of elements in an array: \n ");

scanf (" %d", &num);

printf (" \n Enter %d elements in array: \n ", num);

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

{ printf (" arr[%d] = ", i);

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

}

printf( " Delete the element : \n ");

scanf (" %d", &pos);

if (pos >= num+1)

{

printf (" \n Delete is not possible.");

}

else

{

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

{

arr[i] = arr[i+1];

}

printf (" \n The resultant array is: \n");

for (i = 0; i< num - 1; i++)

{

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

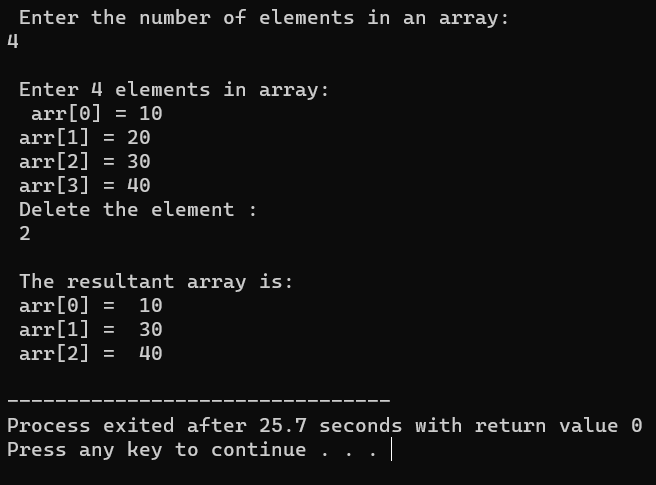
printf (" %d \n", arr[i]);

}

}

return 0;

}



1. Write a C program to check the given string is palindrome or not.

#include <stdio.h>

#include <string.h>

int checkpalindrome(char \*s)

{

int i,c=0,n;

n=strlen(s);

for(i=0;i<n/2;i++)

{

if(s[i]==s[n-i-1])

c++;}

if(c==i)

return 1;

else

return 0;}

int main()

{char s[1000];

printf("Enter the string: ");

gets(s);

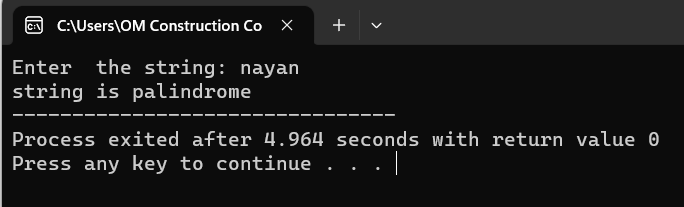
if(checkpalindrome(s))

printf("string is palindrome");

else

printf("string is not palindrome");

}



1. Write a C program to search a particular in the string.

#include <stdio.h>

int main()

{

char str[30],ch;

int ind[10],loop,j;

printf("Enter string: ");

scanf("%[^\n]s",str);

printf("Enter character: ");

getchar();

ch=getchar();

j=0;

for(loop=0; str[loop]!='\0'; loop++)

{

if(str[loop]==ch)

ind[j++]=loop;

}

printf("Input string is: %s\n",str);

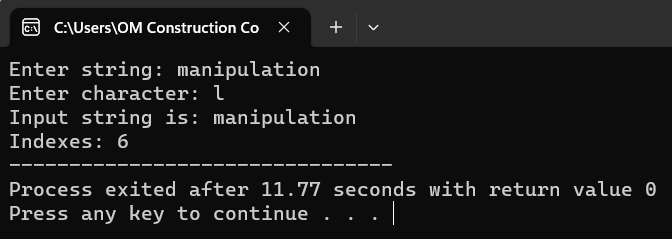
printf("Indexes: ");

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

printf("%d \t",ind[loop]);

return 0;

}



1. Write a C program to perform matrix multiplication

#include<stdio.h>

#include<conio.h>

int main()

{

int mat1[3][3], mat2[3][3], mat3[3][3], sum=0, i, j, k;

printf("Enter first 3\*3 matrix element: ");

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

scanf("%d", &mat1[i][j]);

}

printf("Enter second 3\*3 matrix element: ");

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

scanf("%d", &mat2[i][j]);

}

printf("\nMultiplying two matrices...");

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

{

sum=0;

for(k=0; k<3; k++)

sum = sum + mat1[i][k] \* mat2[k][j];

mat3[i][j] = sum;

}

}

printf("\nMultiplication result of the two given Matrix is: \n");

for(i=0; i<3; i++)

{

for(j=0; j<3; j++)

printf("%d\t", mat3[i][j]);

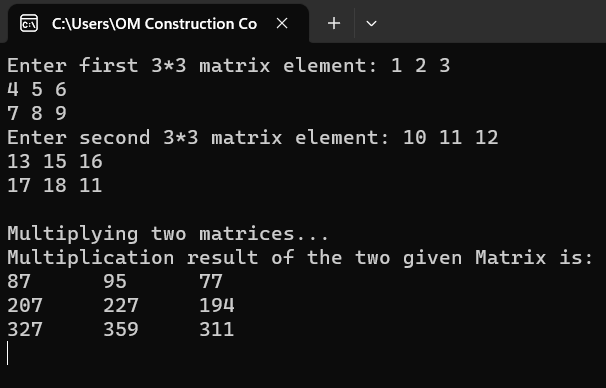
printf("\n");

}

getch();

return 0;

}



1. Write a C program to perform all string manipulation.

#include<stdio.h>

int main()

{

char name[30];

printf("Enter name: ");

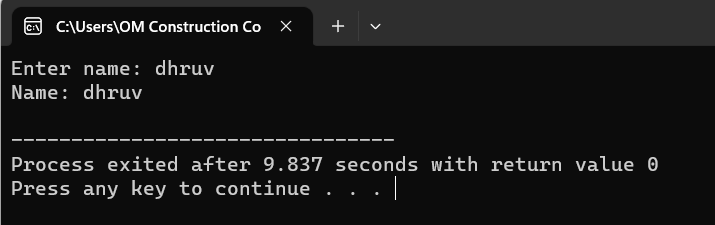
gets(name);

printf("Name: ");

puts(name);

return 0;

}



1. Write a C program to implement single linked list with the following operations

case1: insert an element in the list

case 2: delete an element from the list and display the values

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node\* next;

};

void insert(struct Node\*\* head, int value) {

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

newNode->data = value;

newNode->next = NULL;

if (\*head == NULL) {

\*head = newNode;

} else {

struct Node\* current = \*head;

while (current->next != NULL) {

current = current->next;

}

current->next = newNode;

}

}

void deleteElement(struct Node\*\* head, int value) {

if (\*head == NULL) {

printf("List is empty.\n");

return;

}

struct Node\* current = \*head;

struct Node\* previous = NULL;

while (current != NULL && current->data != value) {

previous = current;

current = current->next;

}

if (current == NULL) {

printf("Element not found in the list.\n");

return;

}

if (previous == NULL) {

\*head = current->next;

} else {

previous->next = current->next;

}

free(current);

printf("Element deleted successfully.\n");

}

void display(struct Node\* head) {

struct Node\* current = head;

while (current != NULL) {

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

current = current->next;

}

printf("\n");

}

int main() {

struct Node\* head = NULL;

int choice, value;

while (1) {

printf("1. Insert an element\n");

printf("2. Delete an element\n");

printf("3. Display the list\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter the value to insert: ");

scanf("%d", &value);

insert(&head, value);

break;

case 2:

printf("Enter the value to delete: ");

scanf("%d", &value);

deleteElement(&head, value);

break;

case 3:

display(head);

break;

case 4:

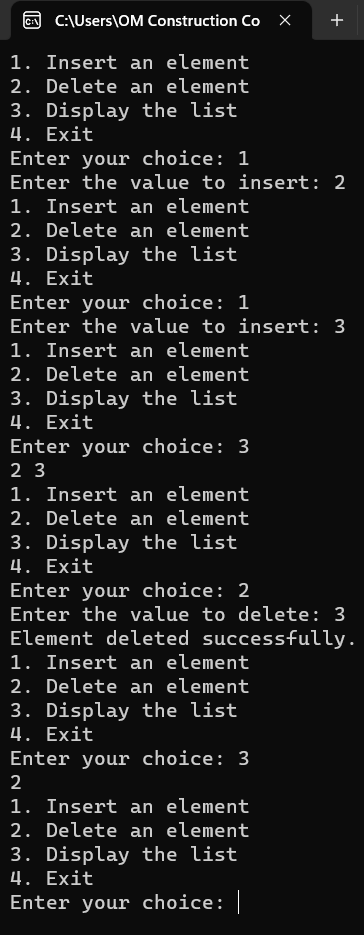
exit(0);

default:

printf("Invalid choice. Please try again.\n");}}

return 0;

}



1. Write a C program to implement stack data structure push an element ,pop an element and display an element from the stack.

#include<stdio.h>

#include<stdlib.h>

#define MAXSIZE 5

struct stack

{

int stk[MAXSIZE];

int top;

};

typedef struct stack ST;

ST s;

void push ()

{

int num;

if (s.top == (MAXSIZE - 1))

{

printf ("Stack is Full\n");

return;

}

else

{

printf ("\nEnter element to be pushed : ");

scanf ("%d", &num);

s.top = s.top + 1;

s.stk[s.top] = num;

}

return;

}

int pop ()

{

int num;

if (s.top == - 1)

{

printf ("Stack is Empty\n");

return (s.top);

}

else

{

num = s.stk[s.top];

printf ("poped element is = %d\n", s.stk[s.top]);

s.top = s.top - 1;

}

return(num);

}

void display ()

{

int i;

if (s.top == -1)

{

printf ("Stack is empty\n");

return;

}

else

{

printf ("\nStatus of elements in stack : \n");

for (i = s.top; i >= 0; i--)

{

printf ("%d\n", s.stk[i]);

}

}

}

int main ()

{

int ch;

s.top = -1;

printf ("\tSTACK OPERATIONS\n");

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

printf(" 1. PUSH\n");

printf(" 2. POP\n");

printf(" 3. DISPLAY\n");

printf(" 4. EXIT\n");

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

while(1)

{

printf("\nChoose operation : ");

scanf("%d", &ch);

switch (ch)

{

case 1:

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

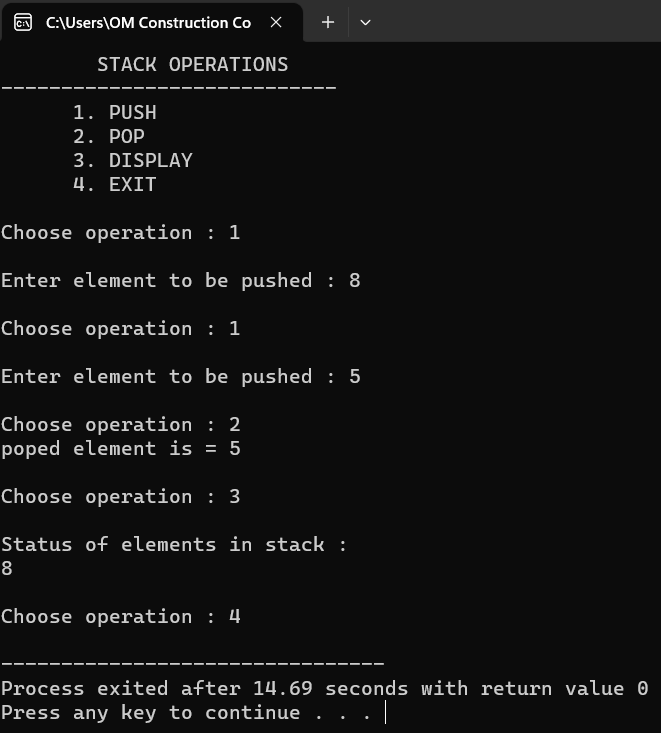
printf("Invalid operation \n");

}

}

return 0;

}



1. Write a C program to implement queue as in follow eneque ,dequeue ,and display

#include<stdio.h>

#define SIZE 3

struct queue

{

int values[SIZE];

int front;

int rear;

};

void enqueue(int);

int dequeue();

int isEmpty();

int isFull();

void display();

struct queue q;

int main()

{

q.front = -1;

q.rear = -1;

int user\_choice, data;

char user\_active = 'Y';

while (user\_active == 'Y' || user\_active == 'y')

{ printf("\n--------Queue Program------\n");

printf("\n1. Enqueue");

printf("\n2. Dequeue");

printf("\n3. Display");

printf("\n4. Exit");

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

scanf("%d", &user\_choice);

switch(user\_choice)

{ case 1:

printf("\nEnter Data: ");

scanf("%d", &data);

enqueue(data);

break;

case 2:

if (!isEmpty())

{

data = dequeue();

printf("\n\* %d was removed!\n", data);}

else

{ printf("\nQueue is Empty!\n");}

break;

case 3:

display();

printf("\n");

break;

case 4:

printf("\n\tProgram was terminated!\n\n");

return 1;

default:

printf("\n\tInvalid Choice\n");

} printf("\nDo You want to continue? ");

fflush(stdin);

scanf(" %c", &user\_active); }

return 0;

} int isEmpty()

{

if (q.front == -1 || q.front > q.rear)

{ return 1; }

return 0; }

int isFull()

{

if (q.rear == SIZE - 1)

{ return 1; }

return 0; }

void enqueue(int data)

{ if (isFull()) {

printf("\nQueue is Full!\n");

return;

} if (isEmpty())

{ q.front += 1; }

q.rear += 1;

q.values[q.rear] = data;

printf("\n\* %d was inserted!\n", data);

} int dequeue()

{ if (!isEmpty())

{ int data = q.values[q.front];

q.front += 1;

return data;}}

void display()

{ if (isEmpty()){

printf("\nQueue is Empty\n");

return;

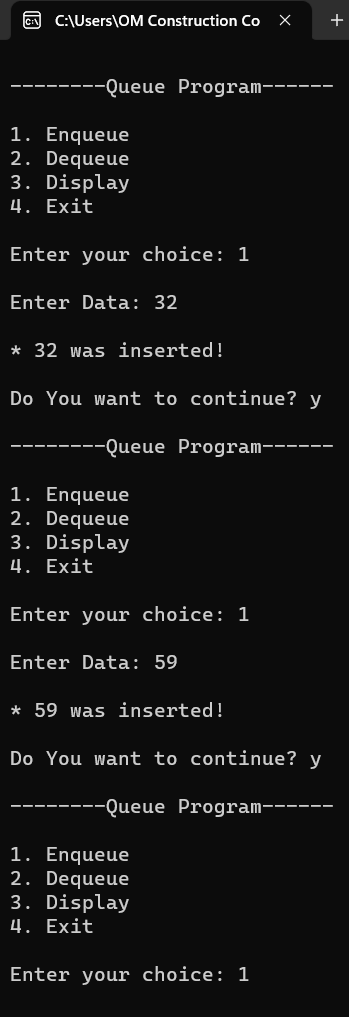
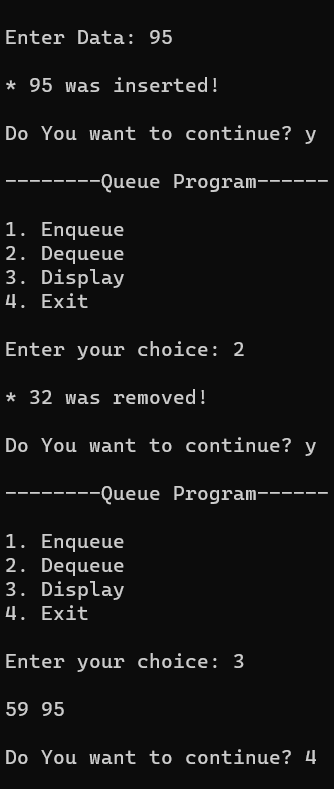
} printf("\n");

int begin = q.front;

while (begin <= q.rear)

{ printf("%d ", q.values[begin]);

begin += 1; }}

1. Write a C program to convert infix expression into postfix using stack.

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{ stack[++top] = x; }

char pop()

{ if(top == -1)

return -1;

else

return stack[top--]; }

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");

e = exp;

while(\*e != '\0')

{ if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{ while((x = pop()) != '(')

printf("%c ", x);}

else

{ while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

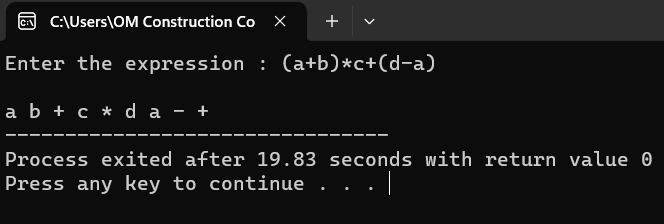
} while(top != -1)

{

printf("%c ",pop());

}return 0;

}



1. Write a C program to evaluate the given expression using stack.

#include <stdio.h>

#include <ctype.h>

#define MAXSTACK 100

#define POSTFIXSIZE 100

int stack[MAXSTACK];

int top = -1;

void push(int item)

{ if (top >= MAXSTACK - 1) {

printf("stack over flow");

return;

}

else {

top = top + 1;

stack[top] = item;

} }

int pop()

{

int item;

if (top < 0) {

printf("stack under flow");

}

else {

item = stack[top];

top = top - 1;

return item;

} } void EvalPostfix(char postfix[])

{ int i;

char ch;

int val;

int A, B;

for (i = 0; postfix[i] != ')'; i++) {

ch = postfix[i];

if (isdigit(ch)) {

push(ch - '0');

}

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/') {

A = pop();

B = pop();

switch (ch)

{

case '\*':

val = B \* A;

break;

case '/':

val = B / A;

break;

case '+':

val = B + A;

break;

case '-':

val = B - A;

break; }

push(val); } }

printf(" \n Result of expression evaluation : %d \n", pop());

}

int main()

{ int i;

char postfix[POSTFIXSIZE];

printf(" There are only four operators(\*, /, +, -) in an expression and operand is single digit only.\n");

printf(" \nEnter postfix expression,\npress right parenthesis ')' for end expression : ");

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

scanf("%c", &postfix[i]);

if (postfix[i] == ')')

{

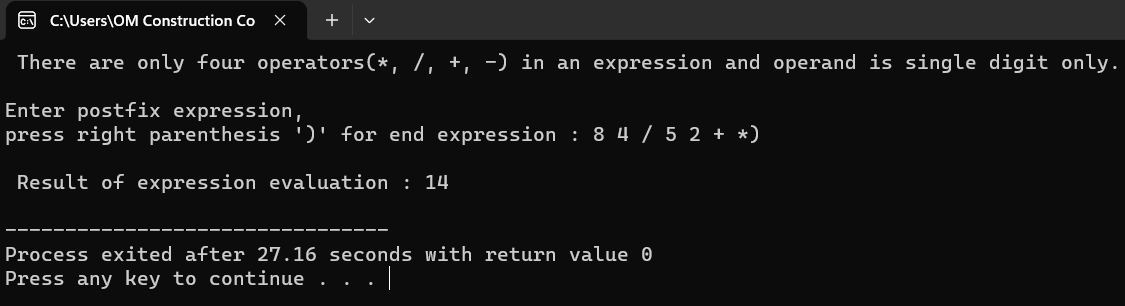
break;

}

} EvalPostfix(postfix);

return 0;

}



1. Write a C program to implement a AVL tree with all rotations.

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int key;

struct Node \*left;

struct Node \*right;

int height;

};

int getHeight(struct Node \*n){

if(n==NULL)

return 0;

return n->height;

}

struct Node \*createNode(int key){

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

node->key = key;

node->left = NULL;

node->right = NULL;

node->height = 1;

return node;

}

int max (int a, int b){

return (a>b)?a:b;

}

int getBalanceFactor(struct Node \* n){

if(n==NULL){

return 0;

}

return getHeight(n->left) - getHeight(n->right);

}

struct Node\* rightRotate(struct Node\* y){

struct Node\* x = y->left;

struct Node\* T2 = x->right;

x->right = y;

y->left = T2;

x->height = max(getHeight(x->right), getHeight(x->left)) + 1;

y->height = max(getHeight(y->right), getHeight(y->left)) + 1;

return x;

}

struct Node\* leftRotate(struct Node\* x){

struct Node\* y = x->right;

struct Node\* T2 = y->left;

y->left = x;

x->right = T2;

x->height = max(getHeight(x->right), getHeight(x->left)) + 1;

y->height = max(getHeight(y->right), getHeight(y->left)) + 1;

return y;

}

struct Node \*insert(struct Node\* node, int key){

if (node == NULL)

return createNode(key);

if (key < node->key)

node->left = insert(node->left, key);

else if (key > node->key)

node->right = insert(node->right, key);

node->height = 1 + max(getHeight(node->left), getHeight(node->right));

int bf = getBalanceFactor(node);

if(bf>1 && key < node->left->key){

return rightRotate(node);

}

if(bf<-1 && key > node->right->key){

return leftRotate(node);

}

if(bf>1 && key > node->left->key){

node->left = leftRotate(node->left);

return rightRotate(node);

}

if(bf<-1 && key < node->right->key){

node->right = rightRotate(node->right);

return leftRotate(node);

}

return node;

}

void preOrder(struct Node \*root)

{

if(root != NULL)

{

printf("%d ", root->key);

preOrder(root->left);

preOrder(root->right);

}

}

int main(){

struct Node \* root = NULL;

root = insert(root, 1);

root = insert(root, 2);

root = insert(root, 4);

root = insert(root, 5);

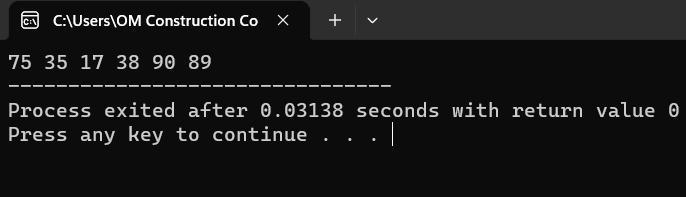
root = insert(root, 6);

root = insert(root, 3);

preOrder(root);

return 0;

}



1. Write a C program to traverse a Binary tree.

#include <stdio.h>

#include <stdlib.h>

struct tnode {

int data;

struct tnode \*left, \*right;

};

struct tnode \*root = NULL;

struct tnode \* createNode(int data) {

struct tnode \*newNode;

newNode = (struct tnode \*) malloc(sizeof(struct tnode));

newNode->data = data;

newNode->left = NULL;

newNode->right = NULL;

return (newNode); }

void insertion(struct tnode \*\*node, int data) {

if (!\*node) {

\*node = createNode(data);

} else if (data < (\*node)->data) {

insertion(&(\*node)->left, data);

} else if (data > (\*node)->data) {

insertion(&(\*node)->right, data); } }

void postOrder(struct tnode \*node) {

if (node) {

postOrder(node->left);

postOrder(node->right);

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

return; }

void preOrder(struct tnode \*node) {

if (node) {

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

preOrder(node->left);

preOrder(node->right); }

return; }

void inOrder(struct tnode \*node) {

if (node) {

inOrder(node->left);

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

inOrder(node->right); }

return; }

int main() {

int data, ch;

while (1) {

printf("\n1. Insertion\n2. Pre-order\n");

printf("3. Post-order\n4. In-order\n");

printf("5. Exit\nEnter your choice:");

scanf("%d", &ch);

switch (ch) {

case 1:

printf("Enter ur data:");

scanf("%d", &data);

insertion(&root, data);

break;

case 2:

preOrder(root);

break;

case 3:

postOrder(root);

break;

case 4:

inOrder(root);

break;

case 5:

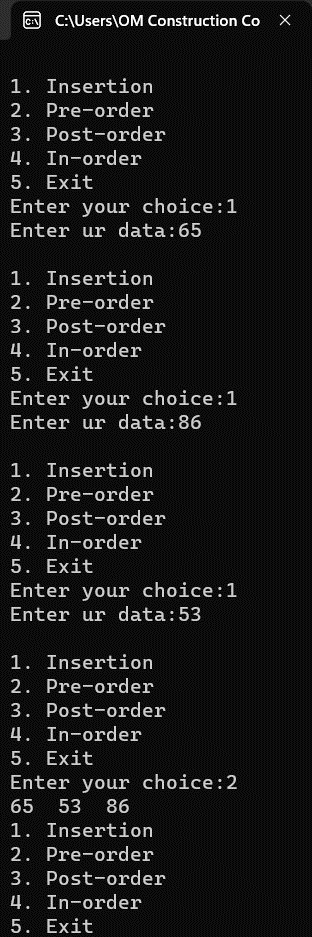
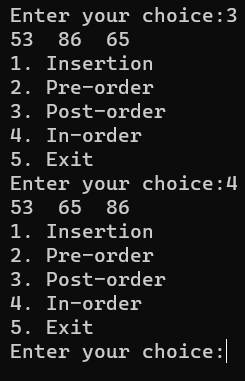
exit(0);

default:

printf("U've entered wrong opetion\n");

break; } }

return 0; }

1. Write a C program to implement hashing using linear probing.

#include <stdio.h>

#include<stdlib.h>

#define TABLE\_SIZE 10

int h[TABLE\_SIZE]={NULL};

void insert()

{

int key,index,i,flag=0,hkey;

printf("\nenter a value to insert into hash table\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE;i++)

{

index=(hkey+i)%TABLE\_SIZE;

if(h[index] == NULL)

{

h[index]=key;

break; } }

if(i == TABLE\_SIZE)

printf("\nelement cannot be inserted\n");

}

void search()

{ int key,index,i,flag=0,hkey;

printf("\nenter search element\n");

scanf("%d",&key);

hkey=key%TABLE\_SIZE;

for(i=0;i<TABLE\_SIZE; i++)

{ index=(hkey+i)%TABLE\_SIZE;

if(h[index]==key)

{

printf("value is found at index %d",index);

break;

}

}

if(i == TABLE\_SIZE)

printf("\n value is not found\n");

}

void display()

{ int i;

printf("\nelements in the hash table are \n");

for(i=0;i< TABLE\_SIZE; i++)

printf("\nat index %d \t value = %d",i,h[i]);

}

main()

{ int opt,i;

while(1)

{

printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");

scanf("%d",&opt);

switch(opt)

{

case 1:

insert();

break;

case 2:

display();

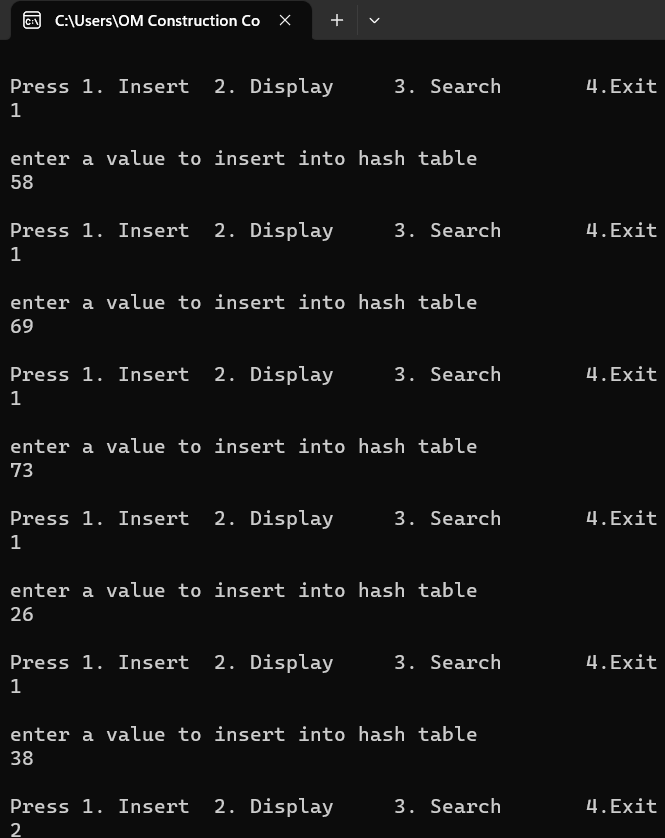
break;

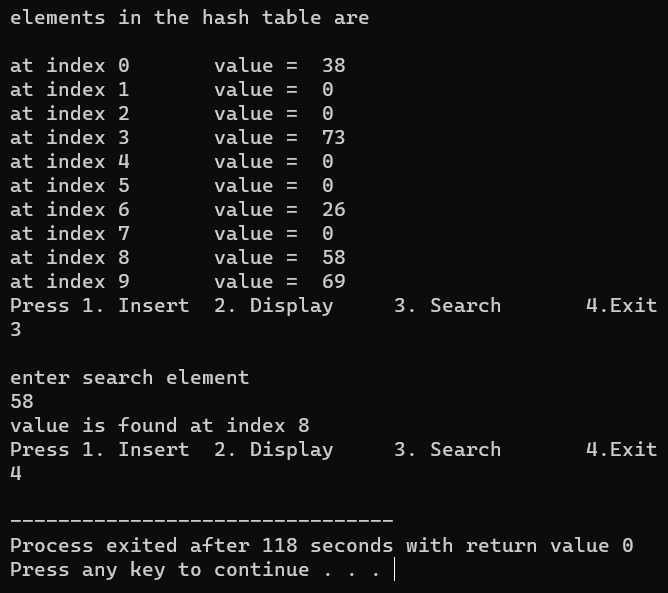
case 3:

search();

break;

case 4:exit(0); } } }





4.

A Write a C program to implement selection sort.

#include <stdio.h>

void selection(int arr[], int n){

int i, j, small;

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

small = i;

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

if (arr[j] < arr[small])

small = j;

int temp = arr[small];

arr[small] = arr[i];

arr[i] = temp;

} }

void printArr(int a[], int n) {

int i;

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

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

}

int main() {

int a[] = { 12, 31, 25, 8, 32, 17 };

int n = sizeof(a) / sizeof(a[0]);

printf("Before sorting array elements are - \n");

printArr(a, n);

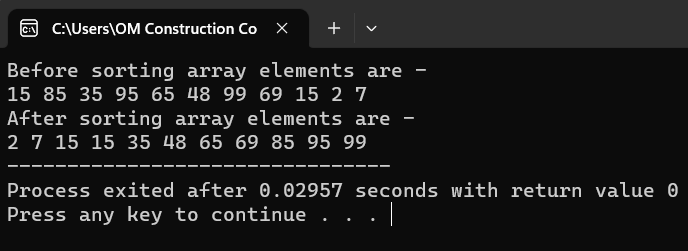
selection(a, n);

printf("\nAfter sorting array elements are - \n");

printArr(a, n);

return 0;

}



B Write a C program to implement insertion sort.

#include <stdio.h>

void insert(int a[], int n)

{ int i, j, temp;

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

temp = a[i];

j = i - 1;

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

{

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

j = j-1;

}

a[j+1] = temp;

}

} void printArr(int a[], int n)

{

int i;

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

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

}

int main()

{

int a[] = { 21, 71, 28, 8, 2, 19 };

int n = sizeof(a) / sizeof(a[0]);

printf("Before sorting array elements are - \n");

printArr(a, n);

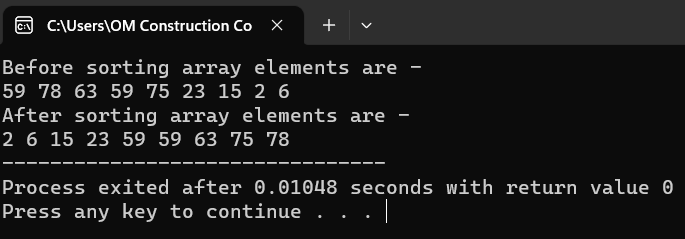
insert(a, n);

printf("\nAfter sorting array elements are - \n");

printArr(a, n);

return 0;

}



c. write a c program to implement quick sort

#include <stdio.h>

void swap(int\* a, int\* b) {

int t = \*a;

\*a = \*b;

\*b = t;

}

int partition(int arr[], int low, int high) {

int pivot = arr[high];

int i = (low - 1);

for (int j = low; j <= high - 1; j++) {

if (arr[j] < pivot) {

i++;

swap(&arr[i], &arr[j]);

}

}

swap(&arr[i + 1], &arr[high]);

return (i + 1);

}

void quickSort(int arr[], int low, int high) {

if (low < high) {

int pi = partition(arr, low, high);

quickSort(arr, low, pi - 1);

quickSort(arr, pi + 1, high);

}

}

void printArray(int arr[], int size) {

int i;

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

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

printf("\n");

}

int main() {

int arr[] = { 12, 17, 6, 25, 1, 5 };

int n = sizeof(arr) / sizeof(arr[0]);

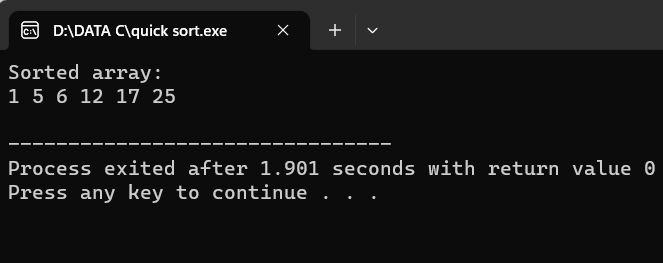
quickSort(arr, 0, n - 1);

printf("Sorted array: \n");

printArray(arr, n);

return 0;

}



d. write a c program to implement Bubble sort

#include <stdio.h>

void bubble\_sort(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]) {

int temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

int main() {

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

int n = sizeof(arr) / sizeof(arr[0]);

bubble\_sort(arr, n);

printf("Sorted array: ");

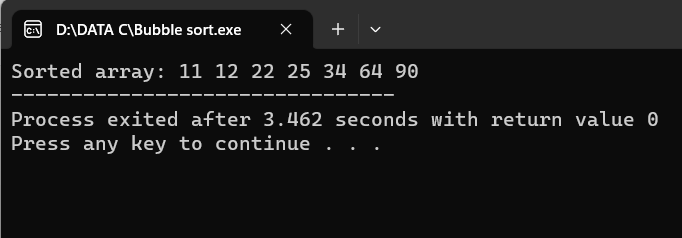
for (int i = 0; i < n; i++) {

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

}

return 0;

}



DAY-5

1.Write a c program to implement single shortest distance.

#include<stdio.h>

#include<conio.h>

#define INFINITY 9999

#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()

{

int G[MAX][MAX],i,j,n,u;

printf("Enter no. of vertices:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

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

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

printf("\nEnter the starting node:");

scanf("%d",&u);

dijkstra(G,n,u);

return 0;

}

void dijkstra(int G[MAX][MAX],int n,int startnode)

{

int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;

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

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

if(G[i][j]==0)

cost[i][j]=INFINITY;

else

cost[i][j]=G[i][j];

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

{

distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n-1)

{

mindistance=INFINITY;

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

if(distance[i]<mindistance&&!visited[i])

{

mindistance=distance[i];

nextnode=i;

}

visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

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

if(i!=startnode)

{

printf("\nDistance of node%d=%d",i,distance[i]);

printf("\nPath=%d",i);

j=i;

do

{

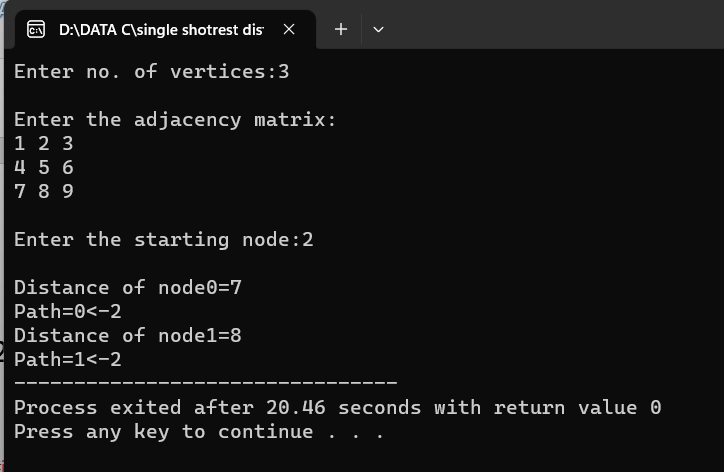
j=pred[j];

printf("<-%d",j);

}while(j!=startnode);

}

}



2.write a c program minimum spamming tree using prims method

#include <stdio.h>

#include <limits.h>

#define V 5

int minKey(int key[], int mstSet[]) {

int min = INT\_MAX, min\_index;

int v;

for (v = 0; v < V; v++)

if (mstSet[v] == 0 && key[v] < min)

min = key[v], min\_index = v;

return min\_index;

}

int printMST(int parent[], int n, int graph[V][V]) {

int i;

printf("Edge Weight\n");

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

printf("%d - %d %d \n", parent[i], i, graph[i][parent[i]]);

} void primMST(int graph[V][V]) {

int parent[V];

int key[V], i, v, count;

int mstSet[V];

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

key[i] = INT\_MAX, mstSet[i] = 0;

key[0] = 0;

parent[0] = -1;

for (count = 0; count < V - 1; count++) {

int u = minKey(key, mstSet);

mstSet[u] = 1;

for (v = 0; v < V; v++)

if (graph[u][v] && mstSet[v] == 0 && graph[u][v] < key[v])

parent[v] = u, key[v] = graph[u][v];

} printMST(parent, V, graph);

} int main() {

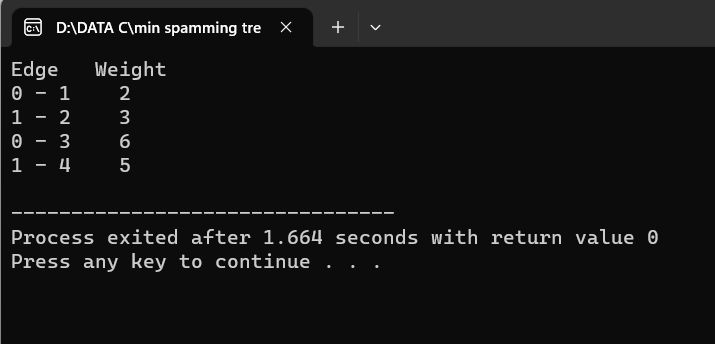
int graph[V][V] = { { 0, 2, 0, 6, 0 }, { 2, 0, 3, 8, 5 },

{ 0, 3, 0, 0, 7 }, { 6, 8, 0, 0, 9 }, { 0, 5, 7, 9, 0 }, };

primMST(graph);

return 0;

}



3.write a c program minimum spamming tree using Kruskal method

#include <stdio.h>

#define MAX 30

typedef struct edge {

int u, v, w;

} edge;

typedef struct edge\_list {

edge data[MAX];

int n;

} edge\_list;

edge\_list elist;

int Graph[MAX][MAX], n;

edge\_list spanlist;

void kruskalAlgo();

int find(int belongs[], int vertexno);

void applyUnion(int belongs[], int c1, int c2);

void sort();

void print();

void kruskalAlgo() {

int belongs[MAX], i, j, cno1, cno2;

elist.n = 0;

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

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

if (Graph[i][j] != 0) {

elist.data[elist.n].u = i;

elist.data[elist.n].v = j;

elist.data[elist.n].w = Graph[i][j];

elist.n++;

}

}

sort();

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

belongs[i] = i;

spanlist.n = 0;

for (i = 0; i < elist.n; i++) {

cno1 = find(belongs, elist.data[i].u);

cno2 = find(belongs, elist.data[i].v);

if (cno1 != cno2) {

spanlist.data[spanlist.n] = elist.data[i];

spanlist.n = spanlist.n + 1;

applyUnion(belongs, cno1, cno2);

}

}

}

int find(int belongs[], int vertexno) {

return (belongs[vertexno]);

}

void applyUnion(int belongs[], int c1, int c2) {

int i;

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

if (belongs[i] == c2)

belongs[i] = c1;

}

void sort() {

int i, j;

edge temp;

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

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

if (elist.data[j].w > elist.data[j + 1].w) {

temp = elist.data[j];

elist.data[j] = elist.data[j + 1];

elist.data[j + 1] = temp;

}

}

void print() {

int i, cost = 0;

for (i = 0; i < spanlist.n; i++) {

printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, spanlist.data[i].w);

cost = cost + spanlist.data[i].w;

}

printf("\nSpanning tree cost: %d", cost);

}

int main() {

int i, j, total\_cost;

n = 6;

Graph[0][0] = 0;

Graph[0][1] = 4;

Graph[0][2] = 4;

Graph[0][3] = 0;

Graph[0][4] = 0;

Graph[0][5] = 0;

Graph[0][6] = 0;

Graph[1][0] = 4;

Graph[1][1] = 0;

Graph[1][2] = 2;

Graph[1][3] = 0;

Graph[1][4] = 0;

Graph[1][5] = 0;

Graph[1][6] = 0;

Graph[2][0] = 4;

Graph[2][1] = 2;

Graph[2][2] = 0;

Graph[2][3] = 3;

Graph[2][4] = 4;

Graph[2][5] = 0;

Graph[2][6] = 0;

Graph[3][0] = 0;

Graph[3][1] = 0;

Graph[3][2] = 3;

Graph[3][3] = 0;

Graph[3][4] = 3;

Graph[3][5] = 0;

Graph[3][6] = 0;

Graph[4][0] = 0;

Graph[4][1] = 0;

Graph[4][2] = 4;

Graph[4][3] = 3;

Graph[4][4] = 0;

Graph[4][5] = 0;

Graph[4][6] = 0;

Graph[5][0] = 0;

Graph[5][1] = 0;

Graph[5][2] = 2;

Graph[5][3] = 0;

Graph[5][4] = 3;

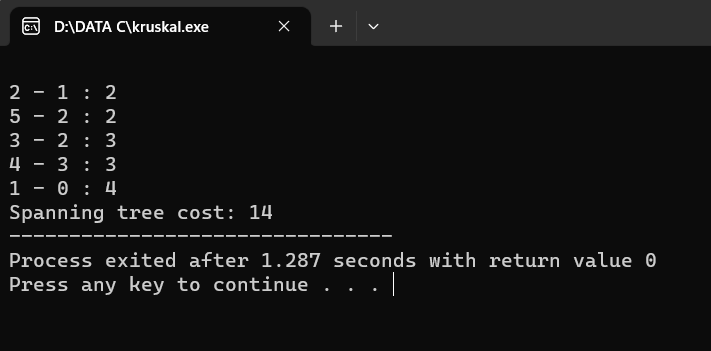
Graph[5][5] = 0;

Graph[5][6] = 0;

kruskalAlgo();

print();

}



4.write a c program to implement depth first search graph traversal

#include <stdio.h>

#include <stdlib.h>

int vis[100];

struct Graph {

int V;

int E;

int\*\* Adj;

};

struct Graph\* adjMatrix()

{ struct Graph\* G = (struct Graph\*)

malloc(sizeof(struct Graph));

if (!G) {

printf("Memory Error\n");

return NULL;

} G->V = 7;

G->E = 7;

G->Adj = (int\*\*)malloc((G->V) \* sizeof(int\*));

for (int k = 0; k < G->V; k++) {

G->Adj[k] = (int\*)malloc((G->V) \* sizeof(int));

}

for (int u = 0; u < G->V; u++) {

for (int v = 0; v < G->V; v++) {

G->Adj[u][v] = 0;

} }

G->Adj[0][1] = G->Adj[1][0] = 1;

G->Adj[0][2] = G->Adj[2][0] = 1;

G->Adj[1][3] = G->Adj[3][1] = 1;

G->Adj[1][4] = G->Adj[4][1] = 1;

G->Adj[1][5] = G->Adj[5][1] = 1;

G->Adj[1][6] = G->Adj[6][1] = 1;

G->Adj[6][2] = G->Adj[2][6] = 1;

return G;

}

void DFS(struct Graph\* G, int u)

{ vis[u] = 1;

printf("%d ", u);

for (int v = 0; v < G->V; v++) {

if (!vis[v] && G->Adj[u][v]) {

DFS(G, v);

}

}

} void DFStraversal(struct Graph\* G)

{ for (int i = 0; i < 100; i++) {

vis[i] = 0;

}

for (int i = 0; i < G->V; i++) {

if (!vis[i]) {

DFS(G, i);

}

}

}

int main()

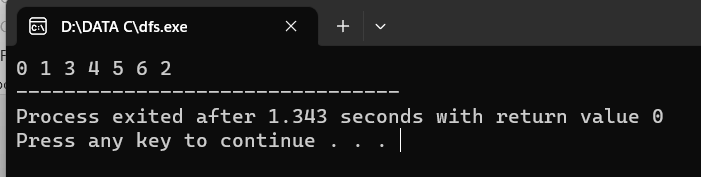
{

struct Graph\* G;

G = adjMatrix();

DFStraversal(G);

}



5.Write a c program to implement breadth first graph traversal

#include <stdio.h>

int n, i, j, visited[10], queue[10], front = -1, rear = -1;

int adj[10][10];

void bfs(int v)

{

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

if (adj[v][i] && !visited[i])

queue[++rear] = i;

if (front <= rear)

{

visited[queue[front]] = 1;

bfs(queue[front++]);

}

}

int main()

{

int v;

printf("Enter the number of vertices: ");

scanf("%d", &n);

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

{

queue[i] = 0;

visited[i] = 0;

}

printf("Enter graph data in matrix form: \n");

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

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

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

printf("Enter the starting vertex: ");

scanf("%d", &v);

bfs(v);

printf("The node which are reachable are: \n");

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

if (visited[i])

printf("%d\t", i);

else

printf("BFS is not possible. Not all nodes are reachable");

return 0;

}

