Exp. No :1.1(a) Date :	**FINDING PRIME NUMBERS IN AN ARRAY**

AIM:

To write a C program to find the prime numbers in an array.

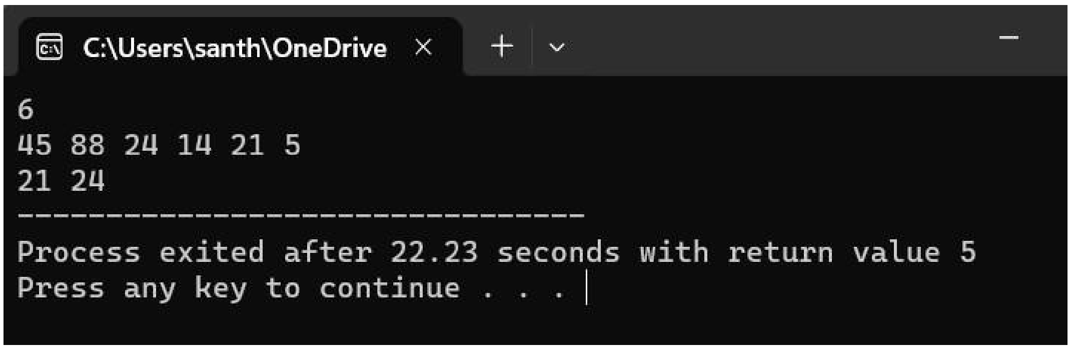
PSEUDOCODE:

```
BEGIN
        Initialize an element
        For (i=0;i<n;i++)
        for (j=i+1;j<n;j++)
        if (a[i]>a[j])
        int t=a[i];
        a[i]=a[j];
        a[j]=t;
        end if
        end for
        for(i=0;i<n;i++)
        if((a[i+1]-a[i])>1)
        c++;
        if(c==x)
        print ("%d",a[i]+1);
        f=1;
        if(f==0)
        print "-1";
        end if
END
```

SOURCE CODE:

```
#include<stdio.h>
int main()
{
int n,x;
scanf("%d %d",&n,&x);
int a[n],i,j;
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(a[i]>a[j])
{
int t=a[i];
a[i]=a[j];
a[j]=t;
}}
}
int c=0,f=0;
for(i=0;i<n;i++)
{
if((a[i+1]-a[i])>1)
{ c++;
if(c==x)
{
printf("%d",a[i]+1); f=1;
break;
}}
}
if(f==0)
printf("-1");
}
```

OUTPUT:



```
6
45 88 24 14 21 5
21 24
--------------------------------
Process exited after 22.23 seconds with return value 5
Press any key to continue . . .
```

RESULT:

Thus, the C program to find the prime numbers in an array is successfully executed and the output is verified**.**

.

Exp. No :1.1(b) Date :	**FIND A MIDDLE ELEMENT INA ARRAY**

AIM:

To write a C-program for finding the middle element in a sorted array.

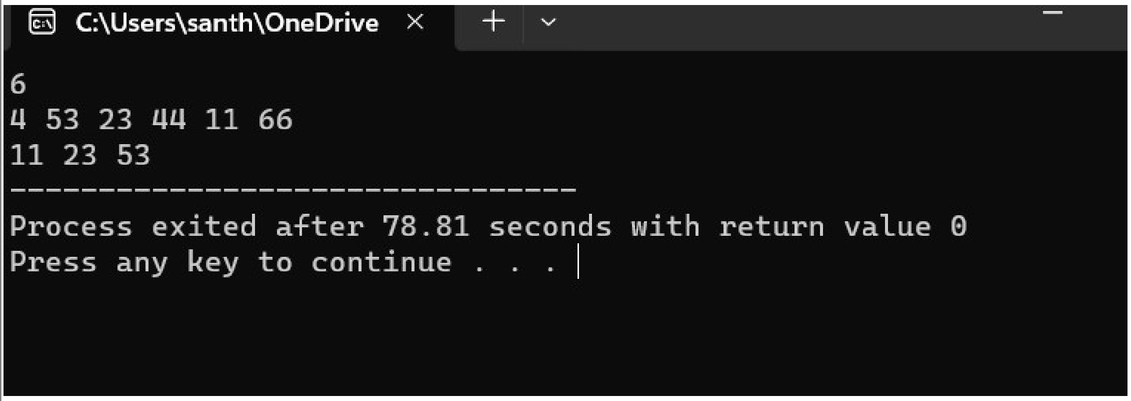
PSEUDOCODE:

```
BEGIN
        Initialize an element
        for(i=0;i<size;i++)
        for(j=i+1;j<size;j++)
        if(arr[i]>arr[j])
         int temp;
        temp=arr[i];
        arr[i]=arr[j];
        arr[j]=temp;
        End if
        End for
        int n=size/2;
        if(size%2==0)
        print ("%d %d",arr[n-1],arr[n]);
        else
        print ("%d",arr[n]);
END
```

SOURCE CODE:

```
#include<stdio.h>
void main()
{
int size,i,j,flag=0;
scanf("%d",&size);
int arr[size];
for(i=0;i<size;i++)
{
scanf("%d",&arr[i]);
}
for(i=0;i<size;i++)
{
 for(j=i+1;j<size;j++)
if(arr[i]>arr[j])
{
int temp;
temp=arr[i];
arr[i]=arr[j];
arr[j]=temp;
}}
int n=size/2;
if(size%2==0){
printf("%d %d",arr[n-1],arr[n]);
}
else{
printf("%d",arr[n]);
}}
```

OUTPUT:



```
6
4 53 23 44 11 66
11 23 53
--------------------------------
Process exited after 78.81 seconds with return value 0
Press any key to continue . . .
```

RESULT:

Thus,the c program to find the middle element in a sorted array is successfully executed and the output is verified**.**

.

.

Exp. No :1.2(a) Date :	**ALPHABETS,DIGITS AND SYMBOLS**

AIM:

To write a C-program to print all the alphabets then digits followed by the symbols.

PSEUDOCODE:

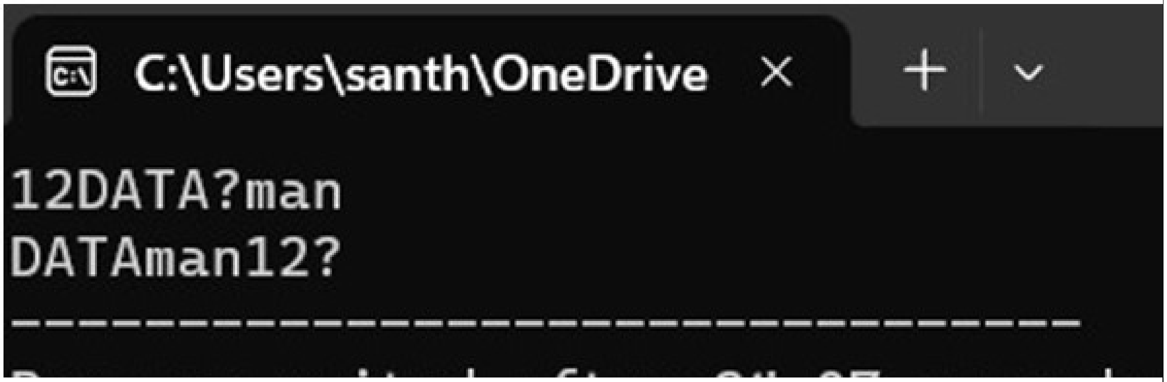
```
BEGIN
        Initializeanelement
        If (isalpha(str[i]))
        printf (str[i]);
        for(i=0;str[i]!='\0';i++)
        if (isdigit(str[i]))
        print(str[i]);
        End if
        End for
        for(i=0;str[i]!='\0';i++)
        if(!isdigit(str[i])&&!isalpha(str[i])) print
        (str[i]);
        End if
        End for
END
```

SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
char str[1000];
int i;
scanf("%s",str);
for(i=0;str[i]!='\0';i++)
{
if(isalpha(str[i]))
printf("%c",str[i]);
}
for(i=0;str[i]!='\0';i++)
{
if(isdigit(str[i]))
```

```
        printf("%c",str[i]);
        }
        for(i=0;str[i]!='\0';i++)
        {
        if(!isdigit(str[i])&&!isalpha(str[i]))
         printf("%c",str[i]);
         }
         return0;
         }
```

OUTPUT:



```
12DATA?man
DATAman12?
---------------------------------
```

RESULT:

Thus, the c program to print all the alphabets then digits followed by the symbols is successfully executed and the output is verified.

Exp. No :1.2(b) Date :	**PRINT THE NON REPEATING CHARACTER**

AIM:

To write a C-program to print then on-repeating characters as output.

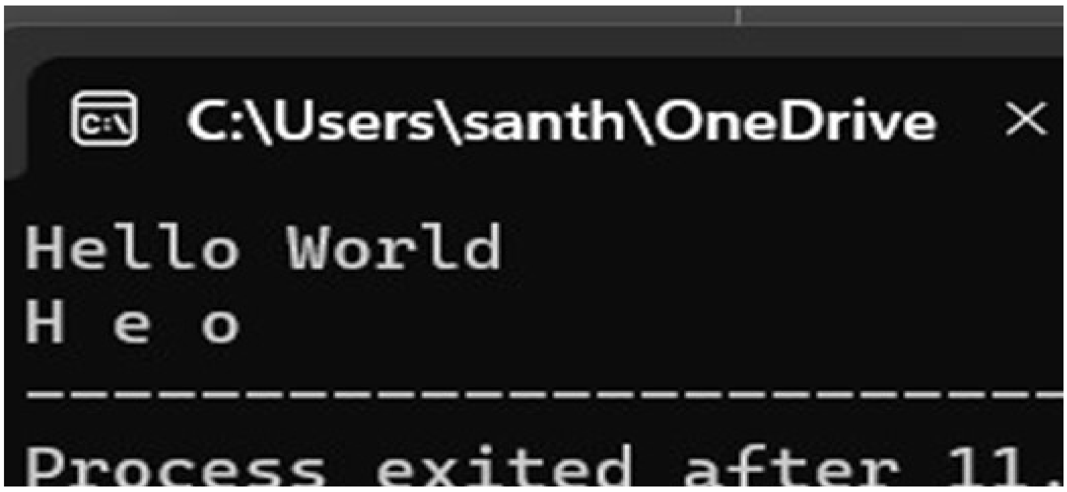
PSEUDOCODE:

```
BEGIN
    Initializean element
    Setfrequencyf[26]={0}
    for(i=0;s[i]!='\0';i++)
    f[s[i]-'a']++
    end for
    for(i=0;s[i]!='\0';i++)
    if(f[s[i]-'a']==1)
    Print (s[i])
    End if
    End for
END
```

SOURCE CODE:

```
#include<stdio.h>
#include<string.h>i
nt main()
{
int i;
char s[100],f[26]={0};
scanf("%s",s);
for(i=0;s[i]!='\0';i++)
{
f[s[i]-'a']++;
   }
   for(i=0;s[i]!='\0';i++)
  {
   if(f[s[i]-'a']==1)
{
printf("%c",s[i]);
}
  }}
```

OUTPUT:



```
C:\Users\santh\OneDrive
Hello World
H e o
--------------------------
Process exited after 11.
```

RESULT:

Thus, the c program to print then on-repeating characters as output is successfully executed and the output is verified.

Exp. No :1.3(a) Date :	**IMPLEMENTATION OF CONCEPT STACK**

AIM:

To write a C-program to the implement the concept of stack data structure.

PSEUDOCODE:

```
BEGIN
        void push(int x){
        if (top==SIZE-1)
        print "Overflow Error"
        else s[++top]=x}
        void pop(){
        if (top==-1)
        print "Underflow"
        else top--}
        int peek(){
        if(top==-1)
        print "Underflow"
        else return s[top]}
        int isFull(){
        if (top==SIZE-1)
        return 1;
        else return 0;}
        int isEmpty(){
        if (top==-1)
        return 1;
        else return 0;}
        int display(){
        for(i=top;i>=0;i--)
        print s[i]}
END
```

SOURCE CODE:

```
        #include<stdio.h>
#include<stdlib.h>
void push();
void pop();
void display();
int isFull();
int isEmpty();
void peek();

int stack[10];
int top=-1,max=3;
```

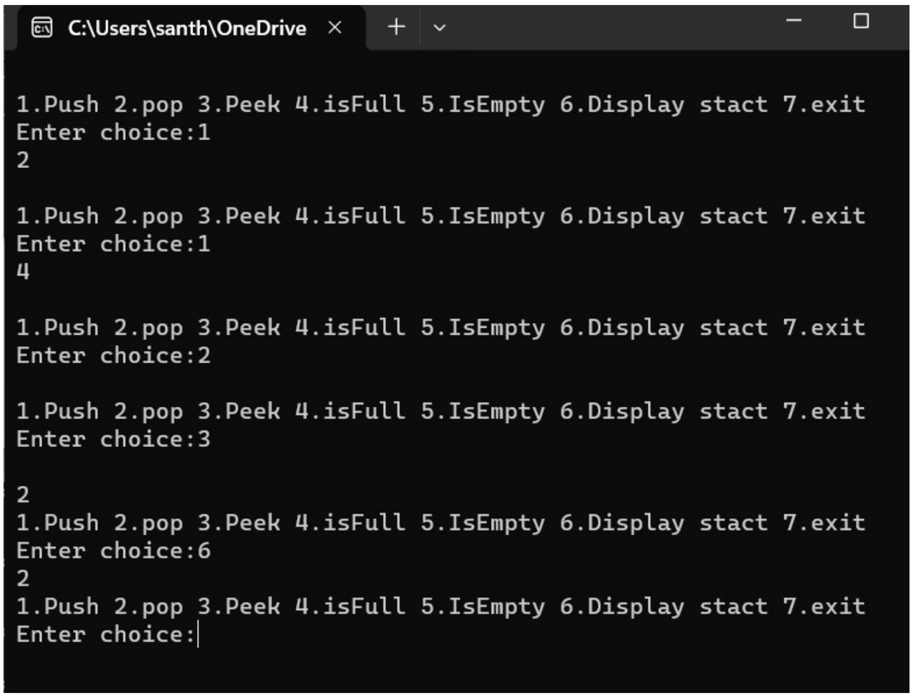
```
void main()
{
	int choice,ele;
	printf("1.Push\n2.Pop\n3.Display\n4.isFull\n5.isEmpty\n6.Peek\n");
	while(1)
	{
		printf("Choice:");
		scanf("%d",&choice);
		switch(choice)
		{
			case 1:
					 push();
					break;
			case 2:
					pop();
					break;
			case 3:
					display();
					break;
			case 4:
					if(isFull())
						printf("Stack is FULL\n");
					else
						printf("Stack is NOT FULL\n");
					break;
			case 5:
					if(isEmpty())
						printf("Stack is EMPTY\n");
					else
						printf("Stack is NOT EMPTY\n");
					break;
			case 6:
					peek();
					break;
			default:
					printf("Program Terminated!!!\n");
					exit(0);
		}
	}
}
void push()
{
	int element;
	if(top==max-1)
		printf("Stack Overflow\n");
	else
	{
		printf("Enter the element: ");
		scanf("%d",&element);
```

```
                stack[++top]=element;
        }

}
void pop()
{
        if(top==-1)
                printf("Stack Underflow\n");
        else
                printf("Popped Element = %d\n",stack[top--]);
}
void display()
{
        int i;
        if(top==-1)
        printf("Stack Empty\n");
        else
        {
                printf("Elements: ");
                for(i=top;i>=0;i--)
                        printf("%d ",stack[i]);
                printf("\n");
        }
}
int isFull()
{
 if(top==max-1)
   return 1;
else
  return 0;
}
int isEmpty()
{
 if(top==-1)
   return 1;
else
  return 0;
}

void peek()
{
        if(top==-1)
                printf("Stack Underflow\n");
        else
                printf("Peek = %d\n",stack[top]);
}
```

OUTPUT:



```
1.Push 2.pop 3.Peek 4.isFull 5.IsEmpty 6.Display stact 7.exit
Enter choice:1
2

1.Push 2.pop 3.Peek 4.isFull 5.IsEmpty 6.Display stact 7.exit
Enter choice:1
4

1.Push 2.pop 3.Peek 4.isFull 5.IsEmpty 6.Display stact 7.exit
Enter choice:2

1.Push 2.pop 3.Peek 4.isFull 5.IsEmpty 6.Display stact 7.exit
Enter choice:3

2
1.Push 2.pop 3.Peek 4.isFull 5.IsEmpty 6.Display stact 7.exit
Enter choice:6
2
1.Push 2.pop 3.Peek 4.isFull 5.IsEmpty 6.Display stact 7.exit
Enter choice:
```

RESULT:

Thus the program of printing the concept of stack is executed and the output is verified successfully.

Exp. No :1.3(b) Date :	**IMPLEMENT BALANCING OF PARANTHESIS**

AIM:

To write a C-program print the implement the balancing of paranthesis using stack data type.

PSEUDOCODE:

```
BEGIN
    for(i=0;a[i]!='\0';i++){
    if((a[i]=='(')||(a[i]=='{')||(a[i]=='[')){
    push(a[i]);}
    else if((a[i]==')')||(a[i]=='}')||(a[i]==']')){
    if((top!=1)&&(((a[i]==')')&&(peek()=='('))||((a[i]=='}')&&(peek()=='{'))||(
    (a[i]==']')&&(peek()=='[')))){pop();}
        else{
            printf("Invalid");}
END
```

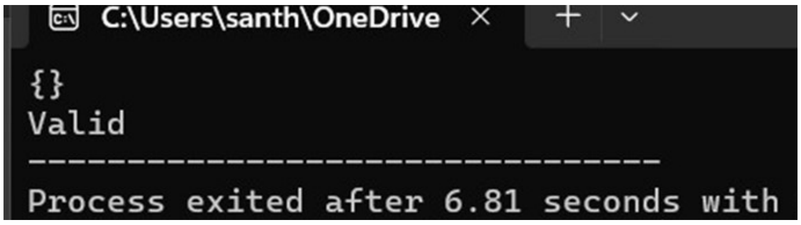
SOURCE CODE:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int check_parentheses(char exp[]);
void push(int);
void pop();
int stack[100];
int top=-1,max=15;
void main()
{
	char exp[50];
	int result,i;
	printf("exp: ");
	scanf("%s",exp);
	result=check_parentheses(exp);
	if(result==0)
		printf("Not balanced\n");
	else
		printf("Balanced\n");
}
```

```
int check_parentheses(char exp[])
{
        int i;
        for(i=0;i<strlen(exp);i++)
        {
                if((exp[i]=='(')||(exp[i]=='{')||(exp[i]=='['))
                        push(exp[i]);
                else if ((exp[i]==')')||(exp[i]=='}')||(exp[i]==']'))
                {

        if(((stack[top]=='(')&&(exp[i]==')'))||((stack[top]=='{')&&(exp[i]=='}'))||((stack[top]=='[')
&&(exp[i]==']')))
                                pop();
                        else {
                        return 0;
                }
        }
        if(top==-1)
                return 1;
                else
                retutn0;
                }Void push(int element){
                stack[top++]=element;
                }Void pop()
                {
                top--;}
```

OUTPUT:



```
{}
Valid
--------------------------------
Process exited after 6.81 seconds with
```

RESULT:

Thus, the C program to print the implement the balancing of paranthesis using stack data type is successfully executed and th output is verified.

.

Exp. No :1.3(c) Date :	**POSTFIX EVALUATION**

AIM:

To write a C-program print the convertion of postfix and evaluation Postfix expression.

PSEUDOCODE:

```
BEGIN
            CONVERSION:
        for ( y[i] != '\0') {
        if (y[i] == '(')
        push(y[i]);
        else if ((y[i] >= 'a' && y[i] <= 'z') || (y[i] >= 'A' && y[i] <= 'Z')) s1[j++] = y[i];
        else if (y[i] == ')') {
        while (peek() != '(') {
        s1[j++] = peek();
        pop();}
        else {
        while (prec(y[i]) <= prec(peek()))
        { s1[j++] = peek();
        pop();} push(y[i]);}}
        while (!isEmpty()) {
         s1[j++] = peek();
        pop();}
        end for
            EVALUATION:
        n2=peek1();
        pop();
        n1=peek1();
        pop();
        switch(y[i]) {
        case  '+':  push1(n1+n2);
        case   '-':  push1(n1-n2);
        case '*':  push1(n1*n2);
        case '/':          push1(n1/n2);
        case '%':    push1(n1%n2);
        }
END
```

SOURCE CODE:

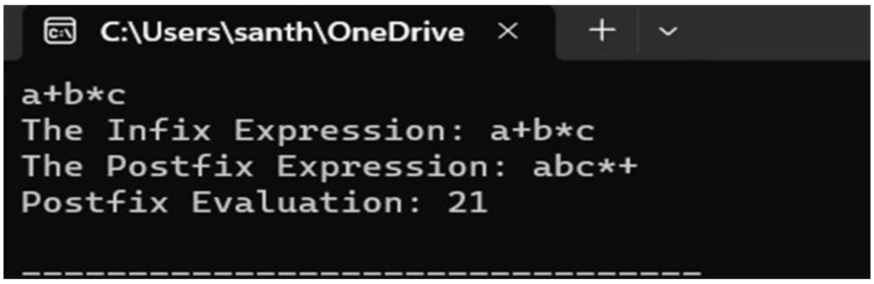
```
#include<stdio.h>
#include<stdlib.h>
    #include<string>
    char [30];
    int d[30];
    char
    s1[30]; int
    TOP = -1;
```

```
void push(char x) {
s[++TOP]=x;}
void push1(int c) {
d[++TOP]=c;}
voidpop() {
TOP--;}
char peek() {
 return s[TOP];}
int peek1() {
return d[TOP];}
            int isEmpty()
            {
            if (TOP ==1)
return1;
else
        return 0;}
int prec(char c) {
if (c == '^')
 return 3;
if (c == '*' || c == '/' || c == '%')
return 2;
if (c == '+' || c == '-')
return 1;
            return 0;}
char *readExp(char *y) {
scanf("%s", y);
            return y;}
char *conversion(char *y) {
int i, j = 0;
for (i = 0; y[i] != '\0'; i++) {
if (y[i] == '(')
            push(y[i]);
else if ((y[i] >= 'a' && y[i] <= 'z') || (y[i] >= 'A' && y[i] <= 'Z'))
s1[j++] = y[i];
else if (y[i] == ')') {
while (peek() != '(') {
s1[j++] = peek();
pop();}
pop();}
else{
while (prec(y[i]) <= prec(peek())) {
s1[j++] = peek();
            pop();}
            push(y[i]);
            }}
            while (!isEmpty()) {
```

```
s1[j++] = peek();
pop();}
TOP = -1;
return s1;}
int evaluation(char *y) {
int i,j;
for(i=0;y[i]!='\0';i++){
if(y[i]=='a'||y[i]=='A')
y[i]='5';
else if(y[i]=='b'||y[i]=='B')
y[i]='8';
else if(y[i]=='c'||y[i]=='C')
y[i]='2';
else if(y[i]=='d'||y[i]=='D')
y[i]='3';
else if(y[i]=='e'||y[i]=='E')
y[i]='6';
else if(y[i]=='f'||y[i]=='F')
y[i]='9';
else if(y[i]=='g'||y[i]=='G')
y[i]='8';}
for(i=0;y[i]!='\0';i++){
if(y[i]>='0' &&
y[i]<='9'){push1(y[i]-
48);}
else {
int n1,n2;
n2=peek1();
pop();
n1=peek1();
pop();
switch(y[i]) {
case '+':
push1(n1+n2);
break;
case '-':
push1(n1-n2);
break;
case '*':
push1(n1*n2);
break;
case '/':
push1(n1/n2);
break;
case '%':
push1(n1%n2);
break;}}}
int v=peek1();
return v;}
int main(void) {
char x[30], y[30];
strcpy(x, readExp(y));
```

```
printf("The Infix Expression: %s\n",
x);strcpy(y, conversion(x));
printf("The Postfix Expression: %s\n", y);
printf("Postfix Evaluation: %d\n",
evaluation(y));return 0;
}
```

OUTPUT:



```
a+b*c
The Infix Expression: a+b*c
The Postfix Expression: abc*+
Postfix Evaluation: 21

--------------------------------
```

RESULT:

Thus, the c program to print the conversion of postfix and evaluation of postfix expression is successfully executed and the output is verified.

Exp. No :1.3(d) Date :	**DELETING THE PAIRED LETTERS**

AIM:

To write a C program to delete as many characters as possible which comes with pairs and printing the resulting string.

PSEUDOCODE:

```
BEGIN
Reduce_string function:
        for (i = 0; i < len; i++)
        {
         if (top >= 0 && s[i] == s[top])
        {
        top--;
        }
        else
        {
        s[++top] = s[i];
        }
        }
        if (top == -1)
        {
        printf("Empty String\n");
        }
        else
        {
        for (i = 0; i <= top; i++)
        {
        printf("%c", s[i]);
        }
        printf("\n");
        }
END
```

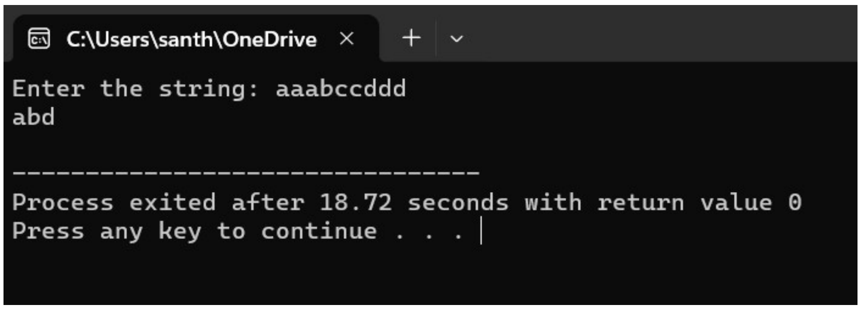
SOURCE CODE:

```
#include <stdio.h>
#include <string.h>

void reduceString(char s[])
{
   int len = strlen(s);
   int top = -1,i;
   for (i = 0; i < len; i++)
        {
```

```
        if (top >= 0 && s[i] == s[top])
                {
            top--;
        }else
                {
            s[++top] = s[i];
        }}
    if (top == -1)
            {
        printf("Empty String\n");
    }
            else{
        for (i = 0; i <= top; i++){
            printf("%c", s[i]);
        }
        printf("\n");
    }
}
 int main()
{
    char s[1001];
    printf("Enter the string: ");
    scanf("%s", s);
    reduceString(s);
    return 0;
}
```

OUTPUT:



```
Enter the string: aaabccddd
abd

--------------------------------
Process exited after 18.72 seconds with return value 0
Press any key to continue . . .
```

RESULT:

Thus, the c program to delete as many characters as possible which comes with pairs and printing the resulting string is successfully executed and the output is verified.