**KPIT TECHNOLOGIES**

**WEEKLY REPORT**

**WEEK 1- Report (DATE: 07/06/2024)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student name** | **Week** | **Branch** | **USN** |
| **C Tejeswar** | **3** | **Circuit (ECE)** | **1NH20EC028** |

**Yashavant Kanetkar Book**

**101-150:**

101. Write a program that extracts part of the given string from the specified

position. For example, if from the sting "Working with strings is fun",

starting from position 3, 4 characters are extracted then it should return

"king".

Sol.

# include <stdio.h>

# include <stdlib.h>

# include <string.h>

int main( )

{

char str[ 20 ], news[ 20 ] ;

char \*s, \*t ;

int pos, n, i ;

printf ( "\nEnter a string: " ) ;

scanf ( "%s", str ) ;

printf ( "Enter position and no. of characters to extract: " ) ;

scanf ( "%d %d", &pos, &n ) ;

s = str ;

t = news ;

if ( pos < 0 || pos > strlen ( str ) )

{

printf ( "Improper position value" ) ;

exit ( 1 ) ;

}

if ( n < 0 )

n = 0 ;

if ( n > strlen ( str ) )

n = n - strlen ( str ) - 1 ;

s = s + pos ;

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

{

\*t = \*s ;

s++ ;

t++ ;

}

\*t = '\0' ;

printf ( "The substring is: %s\n", news ) ;

return 0 ;

}

102. Write a program that converts a string like "124" to an integer 124.

Sol.

include <stdio.h>

int main( )

{

char str[ 6 ] ;

int num = 0, i ;

printf ( "Enter a string containing a number: " ) ;

scanf ( "%s", str ) ;

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

{

if ( str[ i ] >= 48 && str[ i ] <= 57 )

num = num \* 10 + ( str[ i ] - 48 ) ;

else

{

printf ( "Not a valid string\n" ) ;

return 1 ;

}

}

printf ( "The number is: %d\n", num ) ;

return 0 ;

}

103. Write a program that generates and prints the Fibonacci words of order

0 through 5. For example, f(0) = "A", f(1) = "B", f(2) = "BA", f(3) = "BAB",

f(4) = "BABBA", etc.

Sol.

#include <stdio.h>

#include <string.h>

int main( )

{

char str[ 50 ] ;

char lastbutoneterm[ 50 ] = "A" ;

char lastterm[ 50 ] = "B" ;

int i ;

for ( i = 1 ; i <= 5 ; i++ )

{

strcpy ( str, lastterm ) ;

strcat ( str, lastbutoneterm ) ;

printf ( "%s\n", str ) ;

strcpy ( lastbutoneterm, lastterm );

strcpy ( lastterm, str ) ;

}

return 0 ;

}

103. What will be the output of the following programs?

(a) # include <stdio.h>

int main( )

{

char c[ 2 ] = "A" ;

printf ( "%c\n", c[ 0 ] ) ;

printf ( "%s\n", c ) ;

return 0 ;

}

(b) # include <stdio.h>

int main( )

{

char s[ ] = "Get organized! Learn C!!" ;

printf ( "%s\n", &s[ 2 ] ) ;

printf ( "%s\n", s ) ;

printf ( "%s\n", &s ) ;

printf ( "%c\n", s[ 2 ] ) ;

return 0 ;

}

(c) # include <stdio.h>

int main( )

{

char s[ ] = "Borrowers of books spoil the symmetry of shelves" ;

int i = 0 ;

while ( s[ i ] != 0 )

{

printf ( "%c %c\n", s[ i ], \*( s + i ) ) ;

printf ( "%c %c\n", i[ s ], \*( i + s ) ) ;

i++ ;

}

return 0 ;

}

(d) # include <stdio.h>

int main( )

{

char str1[ ] = { ’H’, ’e’, ’l’, ’l’, ’o’, 0 } ;

char str2[ ] = "Hello" ;

printf ( "%s\n", str1 ) ;

printf ( "%s\n", str2 ) ;

return 0 ;

}

(e) # include <stdio.h>

int main( )

{

printf ( 5 + "Good Morning " ) ;

printf ( "%c\n", "abcdefgh"[ 4 ] ) ;

return 0 ;

}

(f) # include <stdio.h>

int main( )

{

printf ( "%d %d %d\n", sizeof ( ’3’ ), sizeof ( "3" ), sizeof ( 3 ) ) ;

return 0 ;

}

Sol.

(a) # include <stdio.h>

int main( )

{

char c[ 2 ] = "A" ;

printf ( "%c\n", c[ 0 ] ) ;

printf ( "%s\n", c ) ;

return 0 ;

}

Output:

A

A

(b) # include <stdio.h>

int main( )

{

char s[ ] = "Get organised! learn C!!" ;

printf ( "%s\n", &s[ 2 ] ) ;

printf ( "%s\n", s ) ;

printf ( "%s\n", &s ) ;

printf ( "%c\n", s[ 2 ] ) ;

return 0 ;

}

Output:

t orgainsed ! learn c !!

Get organised ! learn c !!

et orgainsed ! learn c !!

t

(c) # include <stdio.h>

int main( )

{

char s[ ] = "Borrowers of books spoil the symmetry of shelves" ;

int i = 0 ;

while ( s[ i ] != 0 )

{

printf ( "%c %c\n", s[ i ], \*( s + i ) ) ;

printf ( "%c %c\n", i[ s ], \*( i + s ) ) ;

i++ ;

}

return 0 ;

}

Output:

B B

B B

o o

o o

r r

r r

……

……

s s

s s

(d) # include <stdio.h>

int main( )

{

char str1[ ] = { ’H’, ’e’, ’l’, ’l’, ’o’, 0 } ;

char str2[ ] = "Hello" ;

printf ( "%s\n", str1 ) ;

printf ( "%s\n", str2 ) ;

return 0 ;

}

Output:

Hello

Hello

(e) # include <stdio.h>

int main( )

{

printf ( 5 + "Good Morning " ) ;

printf ( "%c\n", "abcdefgh"[ 4 ] ) ;

return 0 ;

}

Output:

Morning e

(f) # include <stdio.h>

int main( )

{

printf ( "%d %d %d\n", sizeof ( '3' ), sizeof ( "3" ), sizeof ( 3 ) ) ;

return 0 ;

}

Output:

4 2 4

104. If the string "Alice in wonder land" is fed to the following scanf( ) statement, what will be the contents of arrays str1, str2, str3 and str4?

char str1[ 20 ], str2[ 20 ], str3[ 20 ], str4[ 20 ] ;

scanf ( "%s%s%s%s", str1, str2, str3, str4 ) ;

Answer:

str1 – Alice

str2 – in

str3 – wonder

str4 – land

105. To uniquely identify a book a 10-digit ISBN number is used. The rightmost digit in ISBN is a checksum digit. This digit is determined from the other 9 digits using the condition that d1 + 2d2 + 3d3 + ... + 10d10 must be a multiple of 11 (where di denotes the ith digit from the right). The checksum digit d1 can be any value from 0 to 10: the ISBN convention is to use the value X to denote 10. Write a program that receives a 10-digit integer, computes the checksum, and reports whether the ISBN number is correct or not.

Sol.

#include <stdio.h>

#include <string.h>

int main( )

{

char str[ 11 ] ;

int i, j, sum ;

printf ( "Enter 10 digit ISBN number: " ) ;

scanf ( "%s", str ) ;

j = 2 ;

sum = 0 ;

for ( i = 8 ; i >= 0 ; i-- )

{

sum = sum + ( str [ i ] - '0' ) \* j ;

j++ ;

}

for ( i = 0 ; i <= 9 ; i++ )

{

if ( ( sum + i ) % 11 == 0 )

break ;

if ( i == str[ 9 ] - '0' )

printf ( "ISBN Number is verified & found to be correct\n" ) ;

else

printf ( "Checksum error in ISBN Number\n" ) ;

return 0 ;

}

106. A Credit Card number is usually a 16-digit number. A valid Credit Card number would satisfy a rule explained below with the help of a dummy Credit Card number—4567 1234 5678 9129. Start with the rightmost - 1 digit and multiply every other digit by 2.

4 5 6 7 1 2 3 4 5 6 7 8 9 1 2 9

8 12 2 6 10 14 18 4

Then subtract 9 from numbers that are larger than 10. Thus, we get:

8 3 2 6 1 5 9 4

Add them all up to get 38.

Add all the other digits to get 42.

Sum of 38 and 42 is 80. Since 80 is divisible by 10, the Credit Card number is valid.

Write a program that receives a Credit Card number and checks using the above rule whether the Credit Card number is valid.

Sol.

#include <stdio.h>

#include <string.h>

int main( )

{

int len, i, sum, digit, multiple ;

char str[ 20 ] ;

printf ( "Enter the Credit Card number: " ) ;

scanf ( "%s", str ) ;

len = strlen ( str ) ;

sum = 0 ;

for ( i = 15 ; i >= 0 ; i-- )

{

digit = str[ i ] - '0';

if ( i % 2 == 0 )

{

multiple = digit \* 2 ;

digit = multiple < 10 ? multiple : multiple - 9 ;

}

sum += digit ;

}

printf ( "%d\n", sum ) ;

if ( sum % 10 == 0 )

printf ( "Valid credit card number\n" ) ;

else

printf ( "Invalid credit card number\n" ) ;

return 0 ;

}

107. Write a program to store a few strings using an array of pointers to

strings. Receive a string and check if it is present in the array.

Sol.

# include <stdio.h>

# include <string.h>

int main( )

{

char \*str[ ] = {

"We will teach you how to...",

"Move a mountain", "Level a building",

"Erase the past", "Make a million",

"...all through C!"

} ;

char str1[ 20 ], \*p ;

int i ;

printf ( "\nEnter string to be searched: " ) ;

scanf ( "%s", str1 ) ;

p = NULL ;

for ( i = 0 ; i < 6 ; i++ )

{

p = strstr ( str[ i ], str1 ) ;

if ( p != NULL )

{

printf ( "%s found in the array", str1 ) ;

return 0 ;

}

}

printf ( "%s not found in the array", str1 ) ;

return 0 ;

}

108. Write a program to alphabetically sort a set of names stored using an

array of pointers to strings.

Sol.

# include <stdio.h>

# include <string.h>

int main( )

{

char \*str[ ] = {

"Rajesh", "Ashish", "Milind",

"Pushkar", "Akash"

} ;

char \*t ;

int i, j ;

for ( i = 0 ; i < 5 ; i++ )

{

for ( j = i + 1 ; j < 5 ; j++ )

{

if ( ( strcmp ( str[ i ], str[ j ] ) ) > 0 )

{

t = str[ i ] ; str[ i ] = str[ j ] ; str[ j ] = t ;

}

}

}

for ( i = 0 ; i < 5 ; i++ )

printf ( "%s\t", str[ i ] ) ;

return 0 ;

}

109. Write a program to reverse the strings stored in an array of pointers to

strings:

Sol.

# include <stdio.h>

# include <string.h>

void xstrrev ( char \*ss ) ;

int main( )

{

char str[ ][ 35 ] = {

"To ere is human...",

"But to really mess things up...",

"One needs to know C !!"

} ;

int i ;

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

{

xstrrev ( str[ i ] ) ;

printf ( "%s\n", str[ i ] ) ;

}

return 0 ;

}

void xstrrev ( char \*s )

{

int l, i ;

char \*t, temp ;

l = strlen ( s ) ;

t = s + l - 1 ;

for ( i = 1 ; i <= l / 2 ; i++ )

{

temp = \*s ; \*s = \*t ; \*t = temp ;

s++ ; t-- ;

}

}

110. How many bytes in memory would be occupied by the following array of pointers to strings? How many bytes would be required to store the same strings in a two-dimensional character array?

char \*mess[ ] = {

"Hammer and tongs",

"Tooth and nail",

"Spit and polish",

"You and C"

} ;

Answer:

58 bytes for storing strings using array of pointers

68 bytes for storing strings using two-dimensional array

111. Write a program to delete all vowels from a sentence. Assume that the sentence is not more than 80 characters long.

Sol.

Delete all vowels from a sentence

# include <stdio.h>

int main( )

{

char str[ 80 ], str1[ 80 ] ;

char \*s, \*p ;

printf ( "\nEnter a sentence of max 80 characters:\n" ) ;

gets ( str ) ;

s = str ;

p = str1 ;

while ( \*s )

{

if ( \*s == 'a' || \*s == 'e' || \*s == 'i' || \*s == 'o' || \*s == 'u' )

s++ ;

else

if ( \*s == 'A' || \*s == 'E' || \*s == 'I' || \*s == 'O' || \*s == 'U' )

s++ ;

else

\*p++ = \*s++ ;

}

\*p = '\0' ;

printf ( "\n\nSentence after removing all vowels is:\n" ) ;

puts ( str1 ) ;

return 0 ;

}

112. Write a program that will read a line and delete from it all occurrences of the word ‘the’.

Sol.

Delete all occurrences of "the" from a sentence

# include <stdio.h>

int main( )

{

char str[ 80 ], str2[ 80 ] ;

char \*s, \*q, \*p ;

int i ;

printf ( "\nEnter a sentence not more than 80 chars long:\n" ) ;

gets ( str ) ;

s = str ; Base address of the string

p = str2 ; Base address of new string

while ( \*s )

{

q = s ;

if ( \*s == 't' || \*s == 'T' )

{

s++ ;

if ( \*s == 'h' )

{

s++ ;

if ( \*s == 'e' )

;

else

{

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

\*p++ = \*q++ ;

}

}

else

{

\*p++ = \*q++ ;

s-- ;

}

}

else

\*p++ = \*s ;

s++ ;

}

\*p = '\0' ;

printf ( "\nSentence after deleting all occurences of 'the' is:\n" ) ;

puts ( str2 ) ;

return 0 ;

}

113. Write a program that stores a set of names of individuals and abbreviates the first, middle and other names except the last name by their first letter.

Sol.

Convert a full name into initials & last name

# include <stdio.h>

# include <string.h>

int main( )

{

char str1[ 30 ], str2[ 30 ], target[ 30 ] ;

char lastname[ 20 ] ;

char \*p, \*token ;

int count, i, j ;

printf ( "\nEnter name, middle name and surname:\n" ) ;

gets ( str1 ) ;

strcpy ( str2, str1 ) ;

count = 0 ;

token = strtok ( str1, " " ) ;

while ( token != NULL )

{

count++ ;

token = strtok ( NULL, " " ) ;

}

j = 0 ;

i = 0 ;

p = strtok ( str2, " " ) ;

while ( p != NULL )

{

if ( i == count - 1 )

{

strcpy ( lastname, p ) ;

target[ j ] = '\0' ;

}

else

{

target[ j ] = \*p ;

j++ ;

target[ j ] = '.' ;

j++ ;

}

i++ ;

p = strtok ( NULL, " " ) ;

}

strcat ( target, lastname ) ;

printf ( "%s\n", target ) ;

return 0 ;

}

114. Write a program to count the number of occurrences of any two vowels in succession in a line of text. For example, in the sentence

“Please read this application and give me gratuity”

such occurrences are ea, ea, ui.

Sol.

Check for 2 vowels in succession

# include <stdio.h>

int main( )

{

char str[ 80 ] ;

int count = 0 ;

char \*s = str ;

printf ( "\nEnter the string:\n" ) ;

gets ( str ) ;

while ( \*s )

{

if ( \*s == 'a' || \*s == 'e' || \*s == 'i' || \*s == 'o' || \*s == 'u' )

{

s++ ;

if ( \*s == 'a' || \*s == 'e' || \*s == 'i' || \*s == 'o' ||

\*s == 'u' )

count ++ ;

}

s++ ;

}

printf ( "No. of occurrences: %d\n" , count ) ;

return 0 ;

}

115. Write a program that receives an integer (less than or equal to nine digits in length) and prints out the number in words. For example, if the number input is 12342, then the output should be Twelve Thousand Three Hundred Forty Two.

Sol.

Convert number to words

#include<stdio.h>

void convert ( long, char [ ] ) ;

char \*one[ ] = {

" ", " One"," Two"," Three"," Four"," Five",

" Six"," Seven", "Eight"," Nine"," Ten",

" Eleven"," Twelve"," Thirteen"," Fourteen",

"Fifteen"," Sixteen"," Seventeen"," Eighteen",

" Nineteen"

} ;

char \*ten[ ] = {

" ", " ", " Twenty"," Thirty"," Forty"," Fifty",

" Sixty", "Seventy"," Eighty"," Ninety"

} ;

int main( )

{

long num ;

printf ( "\nEnter any Number (max 9 digits): " ) ;

scanf ( "%ld", &num ) ;

if ( num <= 0 )

printf ( "No negative numbers please...\n" ) ;

else

{

convert ( ( num / 10000000 ), "Crore" ) ;

convert ( ( ( num / 100000 ) % 100 ), "Lakh" ) ;

convert ( ( ( num / 1000 ) % 100 ), "Thousand" ) ;

convert ( ( ( num / 100 ) % 10 ), "Hundred" ) ;

convert ( ( num % 100 )," " ) ;

}

}

void convert ( long n, char \*s )

{

if ( n > 19 )

printf ( "%s %s ", ten[ n / 10 ], one[ n % 10 ] ) ;

else

printf ( "%s ", one[ n ] ) ;

if ( n )

printf ( "%s ", s ) ;

}

116: let us write a program that will read a file and count how many characters, spaces, tabs and newlines are present in it.

Sol. # include <stdio.h>

int main( )

{

FILE \*fp ;

char ch ;

int nol = 0, not = 0, nob = 0, noc = 0 ;

fp = fopen ( "PR1.C", "r" ) ;

while ( 1 )

{

ch = fgetc ( fp ) ;

if ( ch == EOF )

break ;

noc++ ;

if ( ch == ' ' )

nob++ ;

if ( ch == '\n' )

nol++ ;

if ( ch == '\t' )

not++ ;

}

fclose ( fp ) ;

printf ( "Number of characters = %d\n", noc ) ;

printf ( "Number of blanks = %d\n", nob ) ;

printf ( "Number of tabs = %d\n", not ) ;

printf ( "Number of lines = %d\n", nol ) ;

return 0 ;

}

OUTPUT:

Number of characters = 125

Number of blanks = 25

Number of tabs = 13

Number of lines = 22

117: Write a program to demonstrate the practical use of these character I/O functions

Sol. # include <stdio.h>

# include <stdlib.h>

int main( )

{

FILE \*fs, \*ft ;

char ch ;

fs = fopen ( "PR1.C", "r" ) ;

if ( fs == NULL )

{

puts ( "Cannot open source file" ) ; exit ( 1 ) ;

}

ft = fopen ( "PR2.C", "w" ) ;

if ( ft == NULL )

{

puts ( "Cannot open target file" ) ;

fclose ( fs ) ; exit ( 2 ) ;

}

while ( 1 )

{

ch = fgetc ( fs ) ;

if ( ch == EOF )

break ;

else

fputc ( ch, ft ) ;

}

fclose ( fs ) ; fclose ( ft ) ;

return 0 ;

}

118: Write a program that writes strings to a file using fputs( ) and then reads them back using fgets( ).

Sol. # include <stdio.h>

# include <stdlib.h>

# include <string.h>

int main( )

{

FILE \*fp ;

char str[ 80 ] ;

fp = fopen ( "POEM.TXT", "w" ) ;

if ( fp == NULL )

{

puts ( "Cannot open file" ) ; exit ( 1 ) ;

}

printf ( "\nEnter a few lines of text:\n" ) ;

while ( strlen ( gets ( str ) ) > 0 )

{

fputs ( str, fp ) ; fputs ( "\n", fp ) ;

}

fclose ( fp ) ;

printf ( "\nFile contents are being read now…\n" , s ) ;

fp = fopen ( "POEM.TXT", "r" ) ;

if ( fp == NULL )

{

puts ( "Cannot open file" ) ; exit ( 2 ) ;

}

while ( fgets ( str, 79, fp ) != NULL )

printf ( "%s" , str ) ;

fclose ( fp ) ;

return 0 ;

}

119: Write a program to record the I/O functions in files.

Sol. # include <stdio.h>

int main( )

{

FILE \*fp ;

struct emp

{

char name[ 40 ] ; int age ; float bs ;

} ;

struct emp e ;

char ch = 'Y' ;

fp = fopen ( "EMPLOYEE.DAT", "w" ) ;

while ( ch == 'Y' )

{

printf ( "Enter name, age, salary: " ) ;

scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;

fprintf ( fp, "%s %d %f\n", e.name, e.age, e.bs ) ;

printf ( "Another record: " ) ;

ch = fgetchar( ) ;

}

fclose ( fp ) ;

fp = fopen ( "EMPLOYEE.DAT", "r" ) ;

while ( fscanf ( fp, "%s %d %f", e.name, &e.age, &e.bs ) != EOF )

printf ( "%s %d %f\n", e.name, e.age, e.bs ) ;

fclose ( fp ) ;

ch = 'Y' ;

fp = fopen ( "EMP.DAT", "wb" ) ;

while ( ch == 'Y' )

{

printf ( "Enter name, age, salary: " ) ;

scanf ( "%s %d %f", e.name, &e.age, &e.bs ) ;

fwrite ( &e, sizeof ( e ), 1, fp ) ;

printf ( "Another record: " ) ;

ch = fgetchar( ) ;

}

fclose ( fp ) ;

fp = fopen ( "EMP.DAT", "rb" ) ;

while ( fread ( &e, sizeof ( e ), 1, fp ) == 1 )

printf ( "%s %d %f\n", e.name, e.age, e.bs ) ;

fclose ( fp ) ;

return 0 ;

}

120: Write a program that performs all the file operations.

Sol. # include <fcntl.h>

# include <sys\types.h>

# include <sys\stat.h>

# include <stdlib.h>

# include <stdio.h>

int main( )

{

char buffer[ 512 ], source[ 128 ], target[ 128 ] ;

int in, out, bytes ;

printf ( "\nEnter source file name: " ) ;

gets ( source ) ;

in = open ( source, O\_RDONLY | O\_BINARY ) ;

if ( in == -1 )

{

puts ( "Cannot open file" ) ; exit ( 1 ) ;

}

printf ( "\nEnter target file name: " ) ;

gets ( target ) ;

out = open ( target, O\_CREAT | O\_BINARY | O\_WRONLY, S\_IWRITE) ;

if ( out == -1 )

{

puts ( "Cannot open file" ) ;

close ( in ) ; exit ( 2 ) ;

}

while ( ( bytes = read ( in, buffer, 512 ) ) > 0 )

write ( out, buffer, bytes ) ;

close ( in ) ; close ( out ) ;

return 0 ;

}

121: Write a program to read a file and display its contents along with line numbers before each line.

Sol. # include <stdio.h>

# include <stdlib.h>

int main( )

{

FILE \*fp ;

char ch, source[ 67 ] ; int count = 1 ;

printf ( "\nEnter file name: " ) ;

scanf ( "%s", source ) ;

fp = fopen ( source, "r" ) ;

if ( fp == NULL )

{

puts ( "Unable to open the file." ) ; exit ( 0 ) ;

}

printf ( "\n%3d: ", count ) ;

while ( ( ch = getc( fp ) ) != EOF )

{

if ( ch == '\n' )

{

count++ ;

printf ( "\n%3d: ", count ) ;

}

else

printf ( "%c", ch ) ;

}

fclose ( fp ) ;

return 0 ;

}

Output:

Enter the file name: Sample.txt

1: What is this life

2: if full of care

3: We have no time

4: to stand and stare!

122: Write a program to append the contents of one file at the end of another.

Sol. # include <stdio.h>

# include <stdlib.h>

# include <string.h>

int main( )

{

FILE \*fs, \*ft ;

char source[ 67 ], target[ 67 ], str[ 80 ] ;

puts ( "Enter source file name: " ) ;

gets ( source ) ;

puts ( "Enter target file name: " ) ;

gets ( target ) ;

fs = fopen ( source, "r" ) ;

if ( fs == NULL )

{

puts ( "Unable to open source file" ) ; exit ( 0 ) ;

}

ft = fopen ( target, "a" ) ;

if ( ft == NULL )

{

fclose ( fs ) ;

puts ( "Unable to open target file" ) ; exit ( 0 ) ;

}

while ( fgets ( str, 79, fs ) != NULL )

fputs ( str, ft ) ;

printf ( "Appending file completed!!" ) ;

fclose ( fs ) ;

fclose ( ft ) ;

return 0 ;

}

Output:

Enter source file name:

Sample.txt

Enter target file name:

NewSample.txt

Appending file completed!!

123: Answer the following questions:

(a) In which file FILE structure is defined?

(b) If a file contains the line “I am a boy\r\n” then on reading this line into the array str[ ] using fgets( ) what would str[ ] contain?

(c) State True or False:

1. The disadvantage of high-level file I/O functions is that the programmer has to manage the file buffers.

2. If a file is opened for reading, it is necessary that the file must exist.

3. If a file opened for writing already exists, its contents would be overwritten.

4. For opening a file in append mode it is necessary that the file should exist.

(d) On opening a file for reading which of the following activities are performed:

1. The disk is searched for existence of the file.

2. The file contents are brought into memory.

3. A pointer is set up which points to the first character in the file.

4. All the above.

(e) Is it necessary that a file created in text mode must always be opened in text mode for subsequent operations?

Sol. (a)The FILE structure is defined in the header file <stdio.h>.

(b) "I am a boy\r\n"

(c) False.

True

True

False

(d) The disk is searched for existence of the file.

A pointer is set up which points to the first character in the file.

(e) No, it is not necessary.

124: Attempt the following questions:

(a) Suppose a file contains student records with each record containing name and age of a student. Write a program to read these records and display them in sorted order by name.

(b) Write a program to copy contents of one file to another. While doing so replace all lowercase characters to their equivalent uppercase characters.

(c) Write a program that merges lines alternately from two files and writes the results to a new file. If one file has a smaller number of lines than the other, the remaining lines from the larger file should be simply copied into the target file.

(d) Write a program to encrypt/decrypt a file using:

(1) Offset cipher: In this cipher each character from the source file is offset with a fixed value and then written to the target file.

For example, if character read from the source file is ‘A’, then write a character represented by ‘A’ + 128 to the target file.

(2) Substitution cipher: In this cipher for each character read from the source file a corresponding predetermined character is written to the target file.

For example, if character ‘A’ is read from the source file, then a ‘!’ would be written to the target file. Similarly, every ‘B’ would be substituted by ‘5’ and so on.

(e) In the file ‘CUSTOMER.DAT’ there are 10 records with the following structure:

struct customer

{

int accno ; char name[ 30 ] ; float balance ;

} ;

In another file ‘TRANSACTIONS.DAT’ there are several records with the following structure:

struct trans

{

int accno ; char trans\_type ; float amount ;

} ;

The element trans\_type contains D/W indicating deposit or withdrawal of amount. Write a program to update ‘CUSTOMER.DAT’ file, i.e., if the trans\_type is ‘D’ then update the balance of ‘CUSTOMER.DAT’ by adding amount to balance for the corresponding accno. Similarly, if trans\_type is ‘W’ then subtract the amount from balance. However, while subtracting the amount ensure that the amount should not get overdrawn, i.e., at least 100 Rs. should remain in the account.

(f) There are 10 records present in a file with the following structure:

struct date { int d, m, y ; } ;

struct employee

{

int empcode[ 6 ] ; char empname[ 20 ] ;

struct date join\_date ; float salary ;

} ;

Write a program to read these records, arrange them in ascending order by join\_date and write them to a target file.

(g) A hospital keeps a file of blood donors in which each record has the format:

Name: 20 columns Address: 40 columns

Age: 2 columns Blood Type: 1 column (Type 1, 2, 3 or 4)

Write a program to read the file and print a list of all blood donors whose age is below 25 and whose blood type is 2.

(h) Given a list of names of students in a class, write a program to store the names in a file on disk. Make a provision to display the nth name in the list, where n is read from the keyboard.

(i) Assume that a Master file contains two fields—roll number and name of the student. At the end of the year, a set of students join the class and another set leaves. A Transaction file contains the roll numbers and an appropriate code to add or delete a student.

Write a program to create another file that contains the updated list of names and roll numbers. Assume that the Master file and the Transaction file are arranged in ascending order by roll numbers.

The updated file should also be in ascending order by roll numbers.

(j) Given a text file, write a program to create another text file deleting the words “a”, “the”, “an” and replacing each one of them with a blank space.

Sol. (a) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_STUDENTS 100

#define NAME\_LEN 30

struct student {

char name[NAME\_LEN];

int age;

};

int compare(const void \*a, const void \*b) {

struct student \*studentA = (struct student \*)a;

struct student \*studentB = (struct student \*)b;

return strcmp(studentA->name, studentB->name);

}

int main() {

struct student students[MAX\_STUDENTS];

int count = 0;

FILE \*file = fopen("students.txt", "r");

if (file == NULL) {

perror("Error opening file");

return EXIT\_FAILURE;

}

while (fscanf(file, "%s %d", students[count].name, &students[count].age) != EOF) {

count++;

}

fclose(file);

qsort(students, count, sizeof(struct student), compare);

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

printf("%s %d\n", students[i].name, students[i].age);

}

return EXIT\_SUCCESS;

}

(b) #include <stdio.h>

#include <ctype.h>

int main() {

char source[128], target[128];

FILE \*srcFile, \*tgtFile;

int ch;

printf("Enter source file name: ");

scanf("%s", source);

printf("Enter target file name: ");

scanf("%s", target);

srcFile = fopen(source, "r");

if (srcFile == NULL) {

perror("Error opening source file");

return EXIT\_FAILURE;

}

tgtFile = fopen(target, "w");

if (tgtFile == NULL) {

perror("Error opening target file");

fclose(srcFile);

return EXIT\_FAILURE;

}

while ((ch = fgetc(srcFile)) != EOF) {

fputc(toupper(ch), tgtFile);

}

fclose(srcFile);

fclose(tgtFile);

return EXIT\_SUCCESS;

}

(c) #include <stdio.h>

int main() {

char file1[128], file2[128], target[128];

FILE \*fp1, \*fp2, \*fpTarget;

char line1[512], line2[512];

printf("Enter first file name: ");

scanf("%s", file1);

printf("Enter second file name: ");

scanf("%s", file2);

printf("Enter target file name: ");

scanf("%s", target);

fp1 = fopen(file1, "r");

if (fp1 == NULL) {

perror("Error opening first file");

return EXIT\_FAILURE;

}

fp2 = fopen(file2, "r");

if (fp2 == NULL) {

perror("Error opening second file");

fclose(fp1);

return EXIT\_FAILURE;

}

fpTarget = fopen(target, "w");

if (fpTarget == NULL) {

perror("Error opening target file");

fclose(fp1);

fclose(fp2);

return EXIT\_FAILURE;

}

while (fgets(line1, sizeof(line1), fp1) != NULL && fgets(line2, sizeof(line2), fp2) != NULL) {

fputs(line1, fpTarget);

fputs(line2, fpTarget);

}

while (fgets(line1, sizeof(line1), fp1) != NULL) {

fputs(line1, fpTarget);

}

while (fgets(line2, sizeof(line2), fp2) != NULL) {

fputs(line2, fpTarget);

}

fclose(fp1);

fclose(fp2);

fclose(fpTarget);

return EXIT\_SUCCESS;

}

(d) #include <stdio.h>

#include <stdlib.h>

void offset\_cipher(const char \*source, const char \*target, int offset) {

FILE \*srcFile, \*tgtFile;

int ch;

srcFile = fopen(source, "r");

if (srcFile == NULL) {

perror("Error opening source file");

exit(EXIT\_FAILURE);

}

tgtFile = fopen(target, "w");

if (tgtFile == NULL) {

perror("Error opening target file");

fclose(srcFile);

exit(EXIT\_FAILURE);

}

while ((ch = fgetc(srcFile)) != EOF) {

fputc(ch + offset, tgtFile);

}

fclose(srcFile);

fclose(tgtFile);

}

int main() {

char source[128], target[128];

int offset = 128;

printf("Enter source file name: ");

scanf("%s", source);

printf("Enter target file name: ");

scanf("%s", target);

offset\_cipher(source, target, offset);

return EXIT\_SUCCESS;

}  
(e) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct customer {

int accno;

char name[30];

float balance;

};

struct trans {

int accno;

char trans\_type;

float amount;

};

int main() {

FILE \*custFile, \*transFile;

struct customer customers[10];

struct trans transaction;

int i, count = 0;

custFile = fopen("CUSTOMER.DAT", "rb+");

if (custFile == NULL) {

perror("Error opening customer file");

return EXIT\_FAILURE;

}

while (fread(&customers[count], sizeof(struct customer), 1, custFile)) {

count++;

}

transFile = fopen("TRANSACTIONS.DAT", "rb");

if (transFile == NULL) {

perror("Error opening transaction file");

fclose(custFile);

return EXIT\_FAILURE;

}

while (fread(&transaction, sizeof(struct trans), 1, transFile)) {

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

if (customers[i].accno == transaction.accno) {

if (transaction.trans\_type == 'D') {

customers[i].balance += transaction.amount;

} else if (transaction.trans\_type == 'W') {

if (customers[i].balance - transaction.amount >= 100) {

customers[i].balance -= transaction.amount;

} else {

printf("Insufficient balance for account %d\n", transaction.accno);

}

}

}

}

}

fclose(transFile);

fseek(custFile, 0, SEEK\_SET);

fwrite(customers, sizeof(struct customer), count, custFile);

fclose(custFile);

return EXIT\_SUCCESS;

}

(f) #include <stdio.h>

#include <stdlib.h>

struct date {

int d, m, y;

};

struct employee {

int empcode[6];

char empname[20];

struct date join\_date;

float salary;

};

int compare\_dates(const void \*a, const void \*b) {

struct employee \*empA = (struct employee \*)a;

struct employee \*empB = (struct employee \*)b;

if (empA->join\_date.y != empB->join\_date.y)

return empA->join\_date.y - empB->join\_date.y;

if (empA->join\_date.m != empB->join\_date.m)

return empA->join\_date.m - empB->join\_date.m;

return empA->join\_date.d - empB->join\_date.d;

}

int main() {

struct employee employees[10];

FILE \*file = fopen("EMPLOYEES.DAT", "r");

if (file == NULL) {

perror("Error opening file");

return EXIT\_FAILURE;

}

for (int i = 0; i < 10; i++) {

fread(&employees[i], sizeof(struct employee), 1, file);

}

fclose(file);

qsort(employees, 10, sizeof(struct employee), compare\_dates);

file = fopen("SORTED\_EMPLOYEES.DAT", "w");

if (file == NULL) {

perror("Error opening file");

return EXIT\_FAILURE;

}

for (int i = 0; i < 10; i++) {

fwrite(&employees[i], sizeof(struct employee), 1, file);

}

fclose(file);

return EXIT\_SUCCESS;

}

(g) #include <stdio.h>

#include <stdlib.h>

struct donor {

char name[20];

char address[40];

int age;

int blood\_type;

};

int main() {

struct donor donor;

FILE \*file = fopen("DONORS.DAT", "r");

if (file == NULL) {

perror("Error opening file");

return EXIT\_FAILURE;

}

while (fread(&donor, sizeof(struct donor), 1, file)) {

if (donor.age < 25 && donor.blood\_type == 2) {

printf("Name: %s\nAddress: %s\nAge: %d\nBlood Type: %d\n", donor.name, donor.address, donor.age, donor.blood\_type);

}

}

fclose(file);

return EXIT\_SUCCESS;

}

(h) #include <stdio.h>

#include <stdlib.h>

int main() {

char names[100][30];

int n, i = 0;

FILE \*file;

file = fopen("students.txt", "w");

if (file == NULL) {

perror("Error opening file");

return EXIT\_FAILURE;

}

printf("Enter student names (type 'end' to stop):\n");

while (1) {

scanf("%s", names[i]);

if (strcmp(names[i], "end") == 0) break;

fprintf(file, "%s\n", names[i]);

i++;

}

fclose(file);

printf("Enter the position of the name to display: ");

scanf("%d", &n);

if (n <= 0 || n > i) {

printf("Invalid position\n");

return EXIT\_FAILURE;

}

file = fopen("students.txt", "r");

if (file == NULL) {

perror("Error opening file");

return EXIT\_FAILURE;

}

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

fgets(names[0], 30, file);

}

printf("The %dth name is: %s", n, names[0]);

fclose(file);

return EXIT\_SUCCESS;

}  
(i) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct student {

int rollno;

char name[30];

};

struct transaction {

int rollno;

char code;

};

int main() {

struct student master[100], updated[100];

struct transaction trans;

int master\_count = 0, updated\_count = 0;

FILE \*masterFile = fopen("MASTER.DAT", "r");

if (masterFile == NULL) {

perror("Error opening master file");

return EXIT\_FAILURE;

}

while (fscanf(masterFile, "%d %s", &master[master\_count].rollno, master[master\_count].name) != EOF) {

master\_count++;

}

fclose(masterFile);

FILE \*transFile = fopen("TRANSACTION.DAT", "r");

if (transFile == NULL) {

perror("Error opening transaction file");

return EXIT\_FAILURE;

}

while (fscanf(transFile, "%d %c", &trans.rollno, &trans.code) != EOF) {

if (trans.code == 'A') {

master[master\_count].rollno = trans.rollno;

strcpy(master[master\_count].name, "New Student");

master\_count++;

} else if (trans.code == 'D') {

for (int i = 0; i < master\_count; i++) {

if (master[i].rollno == trans.rollno) {

for (int j = i; j < master\_count - 1; j++) {

master[j] = master[j + 1];

}

master\_count--;

break;

}

}

}

}

fclose(transFile);

masterFile = fopen("UPDATED\_MASTER.DAT", "w");

if (masterFile == NULL) {

perror("Error opening updated master file");

return EXIT\_FAILURE;

}

for (int i = 0; i < master\_count; i++) {

fprintf(masterFile, "%d %s\n", master[i].rollno, master[i].name);

}

fclose(masterFile);

return EXIT\_SUCCESS;

}

(j) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

int is\_word\_to\_delete(const char \*word) {

return strcmp(word, "a") == 0 || strcmp(word, "the") == 0 || strcmp(word, "an") == 0;

}

void process\_file(const char \*source, const char \*target) {

FILE \*srcFile, \*tgtFile;

char word[128];

srcFile = fopen(source, "r");

if (srcFile == NULL) {

perror("Error opening source file");

exit(EXIT\_FAILURE);

}

tgtFile = fopen(target, "w");

if (tgtFile == NULL) {

perror("Error opening target file");

fclose(srcFile);

exit(EXIT\_FAILURE);

}

while (fscanf(srcFile, "%127s", word) == 1) {

if (!is\_word\_to\_delete(word)) {

fprintf(tgtFile, "%s ", word);

} else {

fprintf(tgtFile, " ");

}

}

fclose(srcFile);

fclose(tgtFile);

}

int main() {

char source[128], target[128];

printf("Enter source file name: ");

scanf("%s", source);

printf("Enter target file name: ");

scanf("%s", target);

process\_file(source, target);

return EXIT\_SUCCESS;

}

125: Write a program where Instead of the program prompting us to enter these filenames, we should be able to supply them at command prompt, in the form: filecopy PR1.C PR2.C

Sol. # include <stdio.h>

# include <stdlib.h>

int main ( int argc, char \*argv[ ] )

{

FILE \*fs, \*ft ;

char ch ;

if ( argc != 3 )

{

puts ( "Improper number of arguments\n" ) ;

exit ( 1 ) ;

}

fs = fopen ( argv[ 1 ], "r" ) ;

if ( fs == NULL )

{

puts ( "Cannot open source file\n" ) ;

exit ( 2 ) ;

}

ft = fopen ( argv[ 2 ], "w" ) ;

if ( ft == NULL )

{

puts ( "Cannot open target file\n" ) ;

fclose ( fs ) ;

exit ( 3 ) ;

}

while ( 1 )

{

ch = fgetc ( fs ) ;

if ( ch == EOF )

break ;

else

fputc ( ch, ft ) ;

}

fclose ( fs ) ;

fclose ( ft ) ;

return 0 ;

}

126: Write a program where there must be a provision to test whether our attempt to read/write was successful or not.

Sol. # include <stdio.h>

int main( )

{

FILE \*fp ;

char ch ;

fp = fopen ( "TRIAL", "w" ) ;

while ( !feof ( fp ) )

{

ch = fgetc ( fp ) ;

if ( ferror( ) )

{

perror ( "TRIAL" ) ;

break ;

}

else

printf ( "%c", ch ) ;

}

fclose ( fp ) ;

return 0 ;

}

127: Write a code to show how we can redirect the output of a program, from the screen to a file.

Sol. C>UTIL.EXE

perhaps I had a wicked childhood,

perhaps I had a miserable youth,

but somewhere in my wicked miserable past,

there must have been a moment of truth ^Z

C>

Or

C>UTIL.EXE > POEM.TXT

C>

128: Answer the following questions:

(a) How will you use the program given below to perform the following operations?

* Copy the contents of one file into another.
* Create a new file and add some text to it.
* Display the contents of an existing file.

# include <stdio.h>

int main( )

{

char ch, str[ 10 ] ;

while ( ( ch = fgetc ( stdin ) ) != -1 )

fputc ( ch, stdout ) ;

return 0 ;

}

(b) State True or False:

1. We can send arguments at command-line even if we define main( ) function without parameters.

2. To use standard file pointers we don’t need to open the file using fopen( ).

3. The zeroth element of argv array points to the name of the executable file.

(c) Write a program using command-line arguments to search for a word in a file and replace it with the specified word. The usage of the program is shown below.

C> change <old word> <new word> <filename>

(d) Write a program that can be used at command prompt as a calculating utility. The usage of the program is shown below.

C> calc <switch> <n> <m> where, n and m are two integer operands and switch is either an

arithmetic operator or a comparison operator. If arithmetic operator is supplied, the output should be the result of the operation. If comparison operator is supplied then the output should be True or False.

Sol. (a) $ ./a.out < source\_file > target\_file

$ ./a.out > new\_file

$ ./a.out < existing\_file

(b) False

True

True

(c) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

void replace\_word\_in\_file(const char \*old\_word, const char \*new\_word, const char \*filename) {

FILE \*file = fopen(filename, "r");

if (!file) {

perror("Error opening file");

exit(EXIT\_FAILURE);

}

FILE \*temp = tmpfile();

if (!temp) {

perror("Error creating temporary file");

fclose(file);

exit(EXIT\_FAILURE);

}

char buffer[1024];

while (fgets(buffer, sizeof(buffer), file)) {

char \*pos;

while ((pos = strstr(buffer, old\_word)) != NULL) {

\*pos = '\0';

fprintf(temp, "%s%s", buffer, new\_word);

strcpy(buffer, pos + strlen(old\_word));

}

fprintf(temp, "%s", buffer);

}

fclose(file);

file = fopen(filename, "w");

if (!file) {

perror("Error reopening file");

fclose(temp);

exit(EXIT\_FAILURE);

}

rewind(temp);

while (fgets(buffer, sizeof(buffer), temp)) {

fputs(buffer, file);

}

fclose(file);

fclose(temp);

}

int main(int argc, char \*argv[]) {

if (argc != 4) {

fprintf(stderr, "Usage: %s <old word> <new word> <filename>\n", argv[0]);

return EXIT\_FAILURE;

}

replace\_word\_in\_file(argv[1], argv[2], argv[3]);

return EXIT\_SUCCESS;

}

(d) #include <stdio.h>

#include <stdlib.h>

#include <string.h>

int main(int argc, char \*argv[]) {

if (argc != 4) {

fprintf(stderr, "Usage: %s <switch> <n> <m>\n", argv[0]);

return EXIT\_FAILURE;

}

char \*operator = argv[1];

int n = atoi(argv[2]);

int m = atoi(argv[3]);

if (strcmp(operator, "+") == 0) {

printf("%d\n", n + m);

} else if (strcmp(operator, "-") == 0) {

printf("%d\n", n - m);

} else if (strcmp(operator, "\*") == 0) {

printf("%d\n", n \* m);

} else if (strcmp(operator, "/") == 0) {

if (m == 0) {

fprintf(stderr, "Error: Division by zero\n");

return EXIT\_FAILURE;

}

printf("%d\n", n / m);

} else if (strcmp(operator, "==") == 0) {

printf("%s\n", (n == m) ? "True" : "False");

} else if (strcmp(operator, "!=") == 0) {

printf("%s\n", (n != m) ? "True" : "False");

} else if (strcmp(operator, "<") == 0) {

printf("%s\n", (n < m) ? "True" : "False");

} else if (strcmp(operator, ">") == 0) {

printf("%s\n", (n > m) ? "True" : "False");

} else if (strcmp(operator, "<=") == 0) {

printf("%s\n", (n <= m) ? "True" : "False");

} else if (strcmp(operator, ">=") == 0) {

printf("%s\n", (n >= m) ? "True" : "False");

} else {

fprintf(stderr, "Invalid operator\n");

return EXIT\_FAILURE;

}

return EXIT\_SUCCESS;

}

129: Write a program that demonstrates the use of >> and << operators:

Sol. # include <stdio.h>

void showbits ( unsigned char ) ;

int main( )

{

unsigned char num = 225, k ;

printf ( "\nDecimal %d is same as binary ", num ) ;

showbits ( num ) ;

k = num >> 1 ;

printf ( "\n%d right shift 1 gives ", num ) ; showbits ( k ) ;

k = num >> 2 ;

printf ( "\n%d right shift 2 gives ", num ) ; showbits ( k ) ;

k = num << 1 ;

printf ( "\n%d left shift 1 gives ", num ) ; showbits ( k ) ;

k = num << 2 ;

printf ( "\n%d left shift 2 gives ", num ) ; showbits ( k ) ;

return 0 ;

}

void showbits ( unsigned char n )

{

int i ;

unsigned char j, k, andmask ;

for ( i = 7 ; i >= 0 ; i-- )

{

j = i ;

andmask = 1 << j ;

k = n & andmask ;

k == 0 ? printf ( "0" ) : printf ( "1" ) ;

}

}

130: Write a program that puts into action both the uses of & operator:

Sol. # include <stdio.h>

void showbits ( unsigned char ) ;

int main( )

{

unsigned char num = 0xAD, j ;

printf ( "\nValue of num = " ) ;

showbits ( num ) ;

j = num & 0x20 ;

if ( j == 0 )

printf ( "\nIts fifth bit is off" ) ;

else

printf ( "\nIts fifth bit is on" ) ;

j = num & 0x08 ;

if ( j == 0 )

printf ( "\nIts third bit is off" ) ;

else

{

printf ( "\nIts third bit is on" ) ;

num = num & 0xF7 ;

printf ( "\nNew value of num = " ) ;

showbits ( num ) ;

j = num & 0x08 ;

if ( j == 0 )

printf ( "\nNow its third bit is turned off" ) ;

}

return 0 ;

}

void showbits ( unsigned char n )

{

int i ;

unsigned char j, k, andmask ;

for ( i = 7 ; i >= 0 ; i-- )

{

j = i ;

andmask = 1 << j ;

k = n & andmask ;

k == 0 ? printf ( "0" ) : printf ( "1" ) ;

}

}

131: The information about colors is to be stored in bits of an unsigned char variable called color. Bit numbers 0 to 6, each represent 7 colors of a rainbow, i.e., bit 0 represents violet, 1 represents indigo, and so on. Write a program that asks the user to enter a number and based on this number it reports which colors in the rainbow do the number represents.

Sol. # include <stdio.h>

# define \_BV(x) ( 1 << x )

void showbits ( unsigned char n );

int main( )

{

unsigned char color, i ;

int c ;

char \*rbcolors[ ] = { "Violet", "Indigo", "Blue", "Green",

"Yellow", "Orange", "Red" } ;

printf ( "\nEnter any number: " ) ;

scanf ( "%d", &c ) ;

color = ( unsigned char ) c ;

printf ( "Colors represented are:\n" ) ;

for ( i = 0 ; i <= 6 ; i++ )

{

if ( ( color & \_BV ( i ) ) == \_BV ( i ) )

printf ( "%s\n", rbcolors[ i ] ) ;

}

return 0 ;

}

Output:

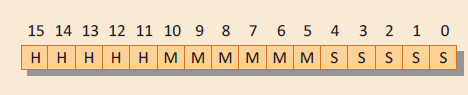
Enter any number: 3

Colors represented are:

Violet

Indigo

132: The time field in a structure is 2 bytes long. Distribution of different bits which account for hours, minutes and seconds is given in Figure 21.6. Define a function that would receive the 2-byte time and print the equivalent hours, minutes and seconds.



Sol. # include <stdio.h>

void display ( unsigned short int time ) ;

int main( )

{

unsigned short int time ;

puts ( "Enter any number less than 24446: " ) ;

scanf ( "%hu", &time ) ;

display ( time ) ;

return 0 ;

}

void display ( unsigned short int tm )

{

unsigned short int hours, minutes, seconds, temp ;

hours = tm >> 11 ;

temp = tm << 5 ;

minutes = temp >> 10 ;

temp = tm << 11 ;

seconds = ( temp >> 11 ) \* 2 ;

printf ( "For Time = %hu\n", tm ) ;

printf ( "Hours = %hu\n", hours ) ;

printf ( "Minutes = %hu\n", minutes ) ;

printf ( "Seconds = %hu\n", seconds ) ;

}

Output:

Enter any number less than 24446:

15500

For Time = 15500

Hours = 7

Minutes = 36

Seconds = 24

133: In an inter-college competition, various sports like cricket, basketball, football, hockey, lawn tennis, table tennis, carom and chess are played between different colleges. The information regarding the games won by a particular college is stored in bit numbers 0, 1, 2, 3, 4, 5, 6, 7 and 8 of an integer variable game. The college that wins in 5 or more than 5 games is awarded the

Champion of Champions trophy. If a number representing the bit pattern mentioned above is entered through the keyboard, then write a program to find out whether the college won the Champion of the Champions trophy or not, along with the names of the games won by the college.

Sol. #include <stdio.h>

int countSetBits(int n) {

int count = 0;

while (n) {

count += n & 1;

n >>= 1;

}

return count;

}

int main() {

int game;

const char \*games[] = {

"Cricket", "Basketball", "Football", "Hockey", "Lawn Tennis",

"Table Tennis", "Carom", "Chess"

};

printf("Enter the number representing the games won (bit pattern): ");

scanf("%d", &game);

int count = countSetBits(game);

if (count >= 5) {

printf("The college won the Champion of Champions trophy!\n");

} else {

printf("The college did not win the Champion of Champions trophy.\n");

}

printf("Games won by the college:\n");

for (int i = 0; i < 8; i++) {

if (game & (1 << i)) {

printf("%s\n", games[i]);

}

}

return 0;

}

134: An animal could be a canine (dog, wolf, fox, etc.), a feline (cat, lynx, jaguar, etc.), a cetacean (whale, narwhal, etc.) or a marsupial (koala, wombat, etc.). The information whether a particular animal is canine, feline, cetacean, or marsupial is stored in bit number 0, 1, 2 and 3, respectively of an integer variable type. Bit number 4 of the variable type stores the information about whether the animal is Carnivore or Herbivore.

For the following animal, complete the program to determine whether the animal is an herbivore or a carnivore. Also determine whether the animal is a canine, feline, cetacean or a marsupial.

struct animal

{

char name[ 30 ] ; int type ;

}

struct animal a = { "OCELOT", 18 } ;

Sol. #include <stdio.h>

struct animal {

char name[30];

int type;

};

int main() {

struct animal a = { "OCELOT", 18 };

const int CANINE = 1 << 0; // Bit 0

const int FELINE = 1 << 1; // Bit 1

const int CETACEAN = 1 << 2; // Bit 2

const int MARSUPIAL = 1 << 3; // Bit 3

const int CARNIVORE = 1 << 4; // Bit 4

if (a.type & CARNIVORE) {

printf("%s is a carnivore.\n", a.name);

} else {

printf("%s is a herbivore.\n", a.name);

}

if (a.type & CANINE) {

printf("%s is a canine.\n", a.name);

}

if (a.type & FELINE) {

printf("%s is a feline.\n", a.name);

}

if (a.type & CETACEAN) {

printf("%s is a cetacean.\n", a.name);

}

if (a.type & MARSUPIAL) {

printf("%s is a marsupial.\n", a.name);

}

return 0;

}

135: In order to save disk space, information about student is stored in an integer variable. Bit numbers 0 to 3 indicate whether the student is a Ist year, IInd year, IIIrd year or IVth year student respectively. Bits 4 to 7 indicate whether the student's stream is Mechanical, Chemical, Electronics or CS. Rest of the bits store room number. Such data for 4 students is stored in the following array: int data[ ] = { 273, 548, 786, 1096 } ; Write a program that uses this data and displays the information about the student.

Sol. #include <stdio.h>

int main() {

int data[] = { 273, 548, 786, 1096 };

const int YEAR\_MASK = 0b1111;

const int STREAM\_MASK = 0b11110000;

const char\* years[] = { "Ist", "IInd", "IIIrd", "IVth" };

const char\* streams[] = { "Mechanical", "Chemical", "Electronics", "CS" };

for (int i = 0; i < 4; ++i) {

int year = data[i] & YEAR\_MASK;

int stream = (data[i] & STREAM\_MASK) >> 4;

int room = data[i] >> 8;

printf("Student %d:\n", i + 1);

printf("Year: %s\n", years[year]);

printf("Stream: %s\n", streams[stream]);

printf("Room Number: %d\n", room);

printf("\n");

}

return 0;

}

136: What will be the output of the following program?

# include <stdio.h>

int main( )

{

int i = 32, j = 65, k, l, m, n, o, p ;

k = i | 35 ; l = ~k ; m = i & j ;

n = j ^ 32 ; o = j << 2 ; p = i >> 5 ;

printf ( "k = %d l = %d m = %d\n", k, l, m ) ;

printf ( "n = %d o = %d p = %d\n", n, o, p ) ;

return 0 ;

}

Sol. k = 51 l = -52 m = 0

n = 97 o = 260 p = 0

137: What is hexadecimal equivalent of each of the following binary numbers?

01011010 11000011

1010101001110101 1111000001011010

Sol. #include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* binaryToHex(char\* binary) {

// Lookup table for hexadecimal digits

char\* hexTable[16] = {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F"};

int binaryLength = strlen(binary);

int paddedLength = (binaryLength % 4 == 0) ? binaryLength : (binaryLength + 4 - (binaryLength % 4));

char\* paddedBinary = (char\*)malloc(paddedLength + 1);

memset(paddedBinary, '0', paddedLength);

paddedBinary[paddedLength] = '\0';

strncpy(paddedBinary + (paddedLength - binaryLength), binary, binaryLength);

int hexLength = paddedLength / 4;

char\* hex = (char\*)malloc(hexLength + 1);

hex[hexLength] = '\0';

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

char group[5];

strncpy(group, paddedBinary + i \* 4, 4);

group[4] = '\0';

int decimal = strtol(group, NULL, 2);

hex[i] = \*hexTable[decimal];

}

free(paddedBinary);

return hex;

}

int main() {

char binary1[] = "01011010";

char binary2[] = "11000011";

char binary3[] = "1010101001110101";

char binary4[] = "1111000001011010";

char\* hex1 = binaryToHex(binary1);

char\* hex2 = binaryToHex(binary2);

char\* hex3 = binaryToHex(binary3);

char\* hex4 = binaryToHex(binary4);

printf("Binary: %s, Hexadecimal: %s\n", binary1, hex1);

printf("Binary: %s, Hexadecimal: %s\n", binary2, hex2);

printf("Binary: %s, Hexadecimal: %s\n", binary3, hex3);

printf("Binary: %s, Hexadecimal: %s\n", binary4, hex4);

free(hex1);

free(hex2);

free(hex3);

free(hex4);

return 0;

}

138: Rewrite the following expressions using bitwise compound assignment operators:

a = a | 3 a = a & 0x48 b = b ^ 0x22 c = c << 2

Sol. a |= 3;

b &= 0x48;

c ^= 0x22;

c <<= 2;

139: Consider an unsigned integer in which rightmost bit is numbered as 0. Write a function checkbits ( x, p, n ) which returns true if all "n" bits starting from position "p" are turned on, false otherwise. For example, checkbits ( x, 4, 3 ) will return true if bits 4, 3 and 2 are 1 in number x.

Sol. #include <stdbool.h>

bool checkbits(unsigned int x, int p, int n) {

unsigned int mask = (1 << n) - 1 << (p - n + 1);

return (x & mask) == mask;

}

int main() {

unsigned int x = 0b11111110;

int p = 4;

int n = 3;

bool result = checkbits(x, p, n);

if (result) {

printf("Bits %d to %d are all turned on in number x.\n", p, p - n + 1);

} else {

printf("Not all bits %d to %d are turned on in number x.\n", p, p - n + 1);

}

return 0;

}

140: Write a program to scan an 8-bit number into a variable and check whether its 3rd, 6th and 7th bit is on.

Sol. #include <stdio.h>

#include <stdbool.h>

int main() {

unsigned char num;

bool bit3, bit6, bit7;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

bit3 = (num & (1 << 2)) != 0;

bit6 = (num & (1 << 5)) != 0;

bit7 = (num & (1 << 6)) != 0;

if (bit3 && bit6 && bit7) {

printf("The 3rd, 6th, and 7th bits are all on.\n");

} else {

printf("The 3rd, 6th, and 7th bits are not all on.\n");

}

return 0;

}

141: Write a program to receive an unsigned 16-bit integer and then exchange the contents of its 2 bytes using bitwise operators.

Sol. #include <stdio.h>

int main() {

unsigned short int num;

printf("Enter an unsigned 16-bit integer: ");

scanf("%hu", &num);

unsigned short int exchanged\_num = ((num & 0xFF) << 8) | ((num >> 8) & 0xFF);

printf("Original number: %hu\n", num);

printf("Exchanged number: %hu\n", exchanged\_num);

return 0;

}

142: Write a program to receive an 8-bit number into a variable and then exchange its higher 4 bits with lower 4 bits.

Sol. #include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

unsigned char exchanged\_num = ((num & 0x0F) << 4) | ((num & 0xF0) >> 4);

printf("Original number: %hhu\n", num);

printf("Exchanged number: %hhu\n", exchanged\_num);

return 0;

}

143: Write a program to receive an 8-bit number into a variable and then set its odd bits to 1.

Sol. #include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

unsigned char modified\_num = num | 0xAA; // 0xAA in binary: 10101010

printf("Original number: %hhu\n", num);

printf("Modified number: %hhu\n", modified\_num);

return 0;

}

144: Write a program to receive an 8-bit number into a variable and then check if its 3rd and 5th bit are on. If these bits are found to be on then put them off.

Sol. #include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

if ((num & (1 << 2)) && (num & (1 << 4))) {

num &= ~(1 << 2);

num &= ~(1 << 4);

printf("3rd and 5th bits were on. They are turned off now.\n");

} else {

printf("3rd and 5th bits were not both on.\n");

}

printf("Original number: %hhu\n", num);

return 0;

}

145: Write a program to receive an 8-bit number into a variable and then check if its 3rd and 5th bit are off. If these bits are found to be off then put them on.

Sol. #include <stdio.h>

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

if (!(num & (1 << 2)) && !(num & (1 << 4))) {

num |= (1 << 2);

num |= (1 << 4);

printf("3rd and 5th bits were off. They are turned on now.\n");

} else {

printf("3rd and 5th bits were not both off.\n");

}

printf("Original number: %hhu\n", num);

return 0;

}

146: Rewrite the showbits( ) function used in this chapter using the \_BV macro.

Sol. #include <stdio.h>

#define \_BV(bit) (1 << (bit))

void showbits(unsigned char num) {

int bit;

for (bit = 7; bit >= 0; bit--) {

if (num & \_BV(bit))

printf("1");

else

printf("0");

}

printf("\n");

}

int main() {

unsigned char num;

printf("Enter an 8-bit number: ");

scanf("%hhu", &num);

printf("Binary representation: ");

showbits(num);

return 0;

}

147: Write a code to force the compiler to explicitly convert the value of an expression to a particular data type.

Sol. # include <stdio.h>

int main( )

{

float a, b ;

int x = 6, y = 4 ;

a = x / y ;

printf ( "Value of a = %f\n", a ) ;

b = ( float ) x / y ;

printf ( "Value of b = %f\n", b ) ;

return 0 ;

}

Output:

Value of a = 1.000000

Value of b = 1.500000

148: Define three functions—fun1( ), fun2( ) and fun3( ). Each function should receive two integers and return a float. Store the addresses of these functions in an array. Call these functions using the addresses stored in the array.

Sol. # include <stdio.h>

float fun1 ( int, int ) ;

float fun2 ( int, int ) ;

float fun3 ( int, int ) ;

float fun1 ( int i, int j )

{

printf ( "In fun1\n" ) ; return 1.0f ;

}

float fun2 ( int i, int j )

{

printf ( "In fun2\n" ) ; return 2.0f ;

}

float fun3 ( int i, int j )

{

printf ( "In fun3\n" ) ; return 3.0f ;

}

int main( )

{

float ( \*ptr[ 3 ] ) ( int, int ) ;

float f ; int i ;

ptr[ 0 ] = fun1 ; ptr[ 1 ] = fun2 ; ptr[ 2 ] = fun3 ;

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

{

f = ( \*ptr[ i ] )( 100, i ) ;

printf ( "%f\n", f ) ;

}

return 0 ;

}

Output:

In fun1

1.000000

In fun2

2.000000

In fun3

3.000000

149: Define a function which can find average of the arguments passed to it. Note that in different calls the function may receive different number of arguments.

Sol. # include <stdio.h>

# include <stdarg.h>

int findavg ( int, ... ) ;

int main( )

{

int avg ;

avg = findavg ( 5, 23, 15, 1, 92, 50 ) ;

printf ( "avg = %d\n", avg ) ;

avg = findavg ( 3, 100, 30, 29 ) ;

printf ( "avg = %d\n", avg ) ;

return 0 ;

}

int findavg ( int tot\_num, ... )

{

int avg, i, num, sum ;

va\_list ptr ;

va\_start ( ptr, tot\_num ) ;

sum = 0 ;

for ( i = 1 ; i <= tot\_num ; i++ )

{

num = va\_arg ( ptr, int ) ;

sum = sum + num ;

}

return ( sum / tot\_num ) ;

}

Output:

avg = 36

avg = 53

150: What will be the output of the following programs?

(a) # include <stdio.h>

int main( )

{

enum status { pass, fail, atkt } ;

enum status stud1, stud2, stud3 ;

stud1 = pass ;

stud2 = fail ;

stud3 = atkt ;

printf ( "%d %d %d\n", stud1, stud2, stud3 ) ;

return 0 ;

}

(b) # include <stdio.h>

int main( )

{

printf ( "%f\n", ( float ) ( ( int ) 3.5 / 2 ) ) ;

printf ( "%d\n", ( int ) ( ( ( float ) 3 / 2 ) \* 3 ) ) ;

return 0 ;

}

Sol. (a) 0 1 2

(b) 1.000000

4