**KPIT TECHNOLOGIES**

**WEEKLY REPORT**

**WEEK 3- Report (DATE: 07/06/2024)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student name** | **Week** | **Branch** | **USN** |
| **M.SAI CHARISHMA** | **3** | **Circuit (ECE)** | **1NH20EC077** |

101. 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

102. 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

103. 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

104.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

105. : 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;

}

106. 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;

}

107. 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;

}

108. : 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;

}

109Write 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;

}

110. 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;

}

111. 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;

}

112. 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;

}

113.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;

114. 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;

}

115. 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

116. 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;

}

117. 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;

}

118: : 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]);

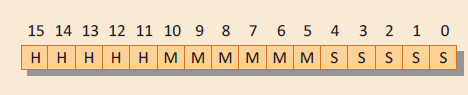
}

}

return 0;

}

119: 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

120The 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

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

# include <stdio.h>

void 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 ) ;

}

Output:

0 1 2

122: # include <stdio.h>

void main()

{

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

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

}

Output:

1.000000

4

123: . Point out the error, if any, in the following programs:

# include <stdio.h>

void main()

{

typedef struct patient

{

char name[ 20 ] ; int age ;

int systolic\_bp ; int diastolic\_bp ;

} ptt ;

ptt p1 = { "anil", 23, 110, 220 } ;

printf ( "%s %d\n", p1.name, p1.age ) ;

printf ( "%d %d\n", p1.systolic\_bp, p1.diastolic\_bp ) ;

}

No Error

124. # include <stdio.h>

void show( ) ;

void main()

{

void ( \*s )( ) ;

s = show ;

( \*s )( ) ;

s( ) ;

}

void show( )

{

printf ( "don't show off. It won't pay in the long run\n" ) ;

}

No Error

; 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, which stores information about a date in a structure containing three members—day, month and year. Using bit fields, the day number should get stored in first 5 bits of day, the month number in 4 bits of month and year in 12 bits of year. Write a program to read date of joining of 10 employees and display them in ascending order of year.

Sol.

# include <stdio.h>

void main()

{

struct date

{

unsigned day : 5 ;

unsigned month : 4 ;

unsigned year : 12 ;

} ;

struct date dt[ 10 ], temp ;

int i, j, d, m, y ;

printf ( "Enter joining dates (dd-mm-yyyy) of 10 employees\n" ) ;

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

{

scanf ( "%d %d %d", &d, &m, &y ) ;

if ( d < 1 || d > 31 || m < 1 || m > 12 )

{

printf ( "Invalid date, enter new date\n" ) ;

i-- ;

continue ;

}

dt[ i ].day = d ;

dt[ i ].month = m ;

dt[ i ].year = y ;

}

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

{

for ( j = i + 1 ; j < 10 ; j++ )

{

if ( dt[ j ].year < dt[ i ].year )

{

temp = dt[ i ] ;

dt[ i ] = dt[ j ] ;

dt[ j ] = temp ;

}

}

}

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

printf ( "%d %d %d\n", dt[ i ].day, dt[ i ].month, dt[ i ].year ) ;

}

126: Write a program to read and store information about insurance policy holder. The information contains details like gender, whether the holder is minor/major, policy name and duration of the policy. Make use of bit-fields to store this information.

Sol.

#include <stdio.h>

# include <string.h>

void main()

{

struct policy\_holder

{

unsigned gender : 1 ; // 0-Male, 1-Female

unsigned status : 1 ; // 0-Minor, 1-Major

char name[ 20 ] ;

unsigned dr : 5 ;

} ;

struct policy\_holder h ;

int g, s, d ;

char n[ 20 ] ; printf ( "\nEnter gender (0-Male, 1-Female): " ) ;

scanf ( "%d", &g ) ;

printf ( "\nEnter status (0-Minor, 1-Major): " ) ;

scanf ( "%d", &s ) ;

printf ( "\nEnter name of the policy holder: " ) ;

scanf ( "%s", n ) ;

printf ( "\nEnter duration (1 to 25 yrs) of the policy: " ) ;

scanf ( "%d", &d ) ;

h.gender = g ;

h.status = s ;

strcpy ( h.name, n ) ;

h.dr = d ;

printf ( "Name: %s\n", h.name ) ;

printf ( "Gender: %s\n", h.gender == 0 ? "Male" : "Female" ) ;

printf ( "Status: %s\n", h.status == 0 ? "Minor" : "Major" ) ;

printf ( "Duration %d\n", h.dr ) ;

}

127: Write a program to calculate the sum of the following series:

1! 2! + 2! 3! + 3! 4! + 4! 5! + …… + 9! 10!

#include <stdio.h>

void main()

{

int i, j ;

float prod1, prod2, term, s ;

s = 0 ;

for ( i = 1 ; i <= 10 ; i++ )

{

prod1 = 1 ;

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

prod1 = prod1 \* j ;

prod2 = prod1 \* j ;

term = prod1 \* prod2 ;

s = s + term ;

}

printf ( "sum of series = %f\n", s ) ;

}

128: Write a program to enter the numbers till the user wants and at the end it should display the count of positive, negative and zeros entered.

#include <stdio.h>

void main()

{

int neg, pos, zero, n ;

char ch = 'y' ;

pos = neg = zero = 0 ;

while ( ch == 'y' || ch == 'Y' )

{

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

scanf ( "%d", &n ) ;

if ( n > 0 )

pos++ ;

if ( n < 0 )

neg++ ;

if ( n == 0 )

zero++ ;

printf ( "Do you want to continue y/n" ) ;

fflush ( stdin ) ;

scanf ( "%c", &ch ) ;

}

printf ( "Positive = %d\n", pos ) ;

printf ( "Negative = %d\n", neg ) ;

printf ( "Zeros = %d\n", zero ) ;

}

129: Write a program to find the range of a set of numbers that are input through the keyboard. Range is the difference between the smallest and biggest number in the list.

#include<stdio.h>

void main()

{

int n, no, flag, small, big ;

flag = 0 ;

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

scanf ( "%d", &n ) ;

while ( n > 0 )

{

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

scanf ( "%d", &no ) ;

if ( flag == 0 )

{

small = big = no ;

flag = 1 ;

}

else

{

if ( no > big )

big = no ;

if ( no < small )

small = no ;

}

n-- ;

}

printf ( "Range: %d", big - small ) ;

}

130: If three integers are entered through the keyboard, write a program to determine whether they form a Pythagorean triplet or not.

#include<stdio.h>

void main()

{

int i, j, k ;

printf ( "Enter three integers: \n" ) ;

scanf ( "%d%d%d", &i, &j, &k ) ;

if ( ( i \* i + j \* j == k \* k ) || ( j \* j + k \* k == i \* i ) ||

( k \* k + i \* i == j \* j ) )

printf ( "Numbers form pythogorean triplet \n" ) ;

else

printf ( "Numbers do not form pythogorean triplet \n" ) ;

}

131: A recursive call should always be subjected to an **if**. Why? Explain with an example.

If the recursive call is not subjected to an **if**, the function would fall in an infinite loop.

void fun( )

{

static int count = 0 ;

count++ ;

if ( count <= 5 )

{

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

fun( ) ;

}

Else

return ;

}

132: Define a function that receives 4 integers and returns sum, product and average of these integers.

#include <stdio.h>

void calc ( int \*, int \*, float \* ) ;

void main()

{

int sum, prod ;

float avg ;

calc ( &sum, &prod, &avg ) ;

printf ( "%d %d %f\n", sum, prod, avg ) ;

}

void calc ( int \*sum, int \*prod, float \*avg )

{

int n1, n2, n3, n4 ;

printf ( "\n Enter four numbers :" ) ;

scanf ( "%d%d%d%d", &n1, &n2, &n3, &n4 ) ;

\*sum = n1 + n2 + n3 + n4 ;

\*prod = n1 \* n2 \* n3 \* n4 ;

\*avg = \*sum / 4.0f ;

}

133. Define a recursive function which prints the prime factors of the number that it receives when called from **main( )** .

#include <stdio.h>

void factor ( int ) ;

void main()

{

int num ;

printf ( "\nEnter a number: " ) ;

scanf ( "%d", &num ) ;

printf ( "\nPrime Factors are: " ) ;

factor ( num ) ;

}

void factor ( int n )

{

static int i = 2 ;

if ( i <= n )

{

if ( n % i == 0 )

{

printf ( "%d ", i ) ;

n = n / i ;

}

else

i++ ;

factor ( n ) ;

}

Return 0 ;

}

134: Write a program that prints sizes of all types of chars, ints and reals.

#include <stdio.h>

void main()

{

char ch ;

unsigned char dh ;

printf ( "character = %d\n", sizeof ( ch ) ) ;

printf ( "unsigned character = %d\n", sizeof ( dh ) ) ;

short int a ;

short unsigned int b ;

int c ;

long int d ;

long unsigned int e ;

printf ( "short signed integer = %d\n", sizeof ( a ) ) ;

printf ( "short unsigned integer = %d\n", sizeof ( b ) ) ;

printf ( "integer = %d\n", sizeof ( c ) ) ;

printf ( "long signed integer = %d\n", sizeof ( d ) ) ;

printf ( "long unsigned integer = %d\n", sizeof ( e ) ) ;

float f ;

double g ;

long double h ;

printf ( "float = %d\n", sizeof ( f ) ) ;

printf ( "double = %d\n", sizeof ( g ) ) ;

printf ( "long double = %d\n", sizeof ( h ) ) ;

}

135How will you allocat space for a **float** array whose size is received from the keyboard?

# include <stdlib.h>

# include <stdio.h>

void main()

{

int sz ;

float \*fptr ;

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

scanf ( "%d", &sz ) ;

fptr = ( float \* ) malloc ( sz \* sizeof ( float ) ) ;

}

# include <stdio.h>

void main()

{

int sz ;

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

scanf ( "%d", &sz ) ;

float arr[ sz ] ;

}

136: Write a function that receives as parameters, a 1-D array, its size and an integer and returns number of times the integer occurs in the array.

#include <stdio.h>

int countnum ( int\*, int, int ) ;

void main()

{

int arr[ 100 ], size, num, i, count ;

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

scanf ( "%d", &size ) ;

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

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

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

printf ( "Enter the number you want to count:\n" ) ;

scanf ( "%d", &num ) ;

count = countnum ( arr, size, num ) ;

printf ( "count = %d", count ) ;

}

int countnum ( int \*a, int sz, int n )

{

int j, cnt = 0 ;

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

{

if ( \*a == n )

cnt++ ;

a++ ;

}

return cnt ;

}

137: Create an array of pointers containing names of 10 cities. Write a program that sorts the cities in reverse alphabetical order and prints this reversed list.

#include <stdio.h>

#include <string.h>

void main()

{

char \*cities[ ] = {

"Nagpur", "Kanpur", "Delhi",

"Sikandarabad", "Akola", "Ghatanji",

"Jabalpur", "Ziri", "Shegaon", "Bombay"

} ;

char \*t ;

int i, j ;

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

{

for ( j = i + 1 ; j < 10 ; j++ )

{

if ( strcmp ( cities[ i ], cities[ j ] ) < 0 )

{

t = cities[ i ] ;

cities[ i ] = cities[ j ] ;

cities[ j ] = t ;

}

}

}

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

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

}

138: . Declare a structure called **student** containing his name, age and address. Create and initialize three structure variables. Define a function to which these variables are passed. The function should

convert the names into uppercase. Print the resultant structure variables.

#include <stdio.h>

#include <string.h>

void upper ( struct stud\* ) ;

struct stud

{

char name[ 20 ] ;

int age ;

char addr[ 40 ] ;

} ;

struct stud s1 = { "akshay", 20, "Ravinagar" } ;

struct stud s2 = { "shubham", 21, "Civil Lines" } ;

struct stud s3 = { "nilesh", 22, "Khamla" } ;

void main()

{

upper ( &s1 ) ;

upper ( &s2 ) ;

upper ( &s3 ) ;

}

void upper ( struct stud \*s )

{

printf ( "Before conversion:\n" ) ;

printf ( "%s %d %s\n", s->name, s->age, s->addr ) ;

strupr ( s->name ) ;

printf ( "After conversion:\n" ) ;

printf ( "%s %d %s\n", s->name, s->age, s->addr ) ;

}

139: Write a program that checks and reports whether sum of elements in the ith row of a 5 x 5 array is equal to sum of elements in ith column.

#include <stdio.h>

void main()

{

int a[ 5 ][ 5 ], i, j, sumr = 0, sumc = 0 ;

printf ( "Enter elements of a 5 x 5 array\n" ) ;

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

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

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

printf ( "Enter row and column you wish to check\n" ) ;

scanf ( "%d", &i ) ;

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

sumr = sumr + a[ i - 1 ][ j ] ;

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

sumc = sumc + a[ j ][ i - 1 ] ;

if ( sumr == sumc )

printf ( "Sums are equal" ) ;

else

printf ( "Sums are not equal" ) ;

}

140: Write a program to multiply two integers using bitwise operators.

#include <stdio.h>

int add ( int, int ) ;

void main()

{

int a, b, result ;

printf ( "\nEnter the numbers to be multiplied :" ) ;

scanf ( "%d%d", &a, &b ) ;

result = 0 ;

while ( b != 0 )

{

if (b & 1 )

result = add ( result, a ) ;

a <<= 1 ;

b >>= 1 ;

}

printf ( "Result:%d", result ) ;

}

int add ( int x, int y )

{

while ( y != 0 )

{

int carry = x & y ;

x = x ^ y ;

y = carry << 1 ;

}

return x ;

}

141: Write a program to count number of words in a given text file.

#include <stdio.h>

#include <stdlib.h>

void main()

{

char ch ; FILE \*fp ;

char fname[ 67 ] ;

int count = 0 ;

printf ( "Enter File name: " ) ;

gets ( fname ) ;

fp = fopen ( fname, "r" ) ;

if ( fp == NULL )

{

printf ("Unable to open file\n" ) ;

exit ( 1 ) ;

}

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

{

if ( ch == ' ' )

count++ ;

}

printf ( "No of words = %d", count + 1 ) ;

fclose ( fp ) ;

}

142: Write a program that receives a set of numbers as command- line arguments and prints their average.

#include <stdio.h>

#include <stdlib.h>

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

{

int sum, i, avg ;

sum = 0 ;

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

sum = sum + atoi ( argv[ i ] ) ;

avg = sum / ( argc - 1 ) ;

printf ( "Average = %d", avg ) ;

}

143: Write a program to check whether contents of the two files are same by comparing them on a byte-by-byte basis.

#include <stdio.h>

#include <stdlib.h>

void main()

{

char ch1,ch2 ;

FILE \*fp,\*fq ;

char fname1[ 67 ], fname2[ 67 ] ;

printf ( "Enter File1 name: " ) ;

gets ( fname1 ) ;

printf ( "Enter File2 name: " ) ;

gets ( fname2 ) ;

fp = fopen ( fname1, "rb" ) ;

if ( fp == NULL )

{

printf ("Unable to open file1\n" ) ;

exit ( 1 ) ;

}

fq = fopen ( fname2, "rb" ) ;

if ( fq == NULL )

{

printf ("Unable to open file2\n" ) ;

exit ( 1 ) ;

}

while ( 1 )

{

ch1 = getc ( fp ) ;

ch2 = getc ( fq ) ;

if ( ch1 == EOF && ch2 == EOF )

{

printf ( "File contents match" ) ;

break ;

}

if ( ch1 != ch2 || ch1 == EOF && ch2 != EOF ||

ch1 != EOF && ch2 == EOF )

{

printf ( "File contents do not match" ) ;

break ;

}

}

fclose ( fp ) ;

fclose ( fq ) ;

}

144: Dynamically allocate space for a 3-D array of dimensions 3 x 5 x 4. Set up each elements of this array with a value 10. Report an error, if enough memory space is not available.

#include <stdio.h>

#include <stdlib.h>

void main()

{

int i, j, k ;

int \*a ;

a = ( int \* ) malloc ( 3 \* 5 \* 4 \* sizeof ( int ) ) ;

if ( a == NULL )

printf ( "Insufficient space\n" ) ;

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

{

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

{

for ( k = 0 ; k < 4 ; k++ )

{

\*( a + i \* 5 \* 4 + j \* 4 + k ) = 10 ;

printf ( "%d ", \*( a + i \* 5 \* 4 + j \* 4 + k ) = 10 ) ;

}

printf ( "\n" ) ;

}

printf ( "\n\n" ) ;

}

}

145: Point out errors, in any, in the following code snippet:

int a ;

float b ;

char ch ;

scanf ( "%d %f %c", a, b, ch ) ;

printf ( "%d %f %c", &a, &b, &ch ) ;

Error. “&” should be used before each variable in **scanf( )**.

146: Write a program that defines a function called **isalnum( )**. The function should receive a string and check if all characters in it are alphabets or digits. If so, it should return a true, otherwise false. Call this function for the following strings:

"ABCD1234"

"Nagpur – 440010"

#include <stdio.h>

#define TRUE 1

#define FALSE 0

int isalnum ( char \* ) ;

void main()

{

char str1[ ] = "ABCD1234" ;

char str2[ ] = "Nagpur – 440010" ;

int ret ;

ret = isalnum ( str1 ) ;

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

ret = isalnum ( str2 ) ;

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

}

int isalnum ( char \*p )

{

while ( \*p )

{

if ( \*p >= '0' && \*p <= '9' || \*p >= 'a' && \*p <= 'z' ||

\*p >= 'A' && \*p <= 'Z' )

p++ ;

else

return FALSE ;

}

return TRUE ;

}

147: Define an enumeration to represent colors red, green, yellow, magenta and brown. Create two variables **Apple** and **Banana** of this **enum** type and assign colors red and yellow to them respectively. Print these color values and indicate what output will they produce.

#include <stdio.h>

#include <stdlib.h>

void main()

{

enum fruits { red, green, yellow, magenta, brown } ;

enum fruits Apple, Banana ;

Apple = red ;

Banana = yellow ;

printf ( "%d %d\n", Apple, Banana ) ;

}

Output

0 2

148: Define a function called **showbits( )** which displays all the bits of an **unsigned char** that it receives. Call this function for values 45 and 30. Indicate what output will **showbits( )** produce for these values?

# include <stdio.h>

void showbits ( unsigned char ) ;

void main()

{

unsigned char num1 = 45, num2 = 30 ;

showbits ( num1 ) ;

showbits ( num2 ) ;

}

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" ) ;

}

printf ( "\n" ) ;

}

149: Write a program to generate and print all unique combinations of numbers 1, 2, 3 and 4.

#include <stdio.h>

void main()

{

int i, j, k, l ;

for ( i = 1 ; i <= 4 ; i++ )

{

for ( j = 1 ; j <= 4 ; j++ )

{

for ( k = 1 ; k <= 4 ; k++ )

{

for ( l = 1 ; l <= 4 ; l++ )

{

if ( i != j && i != k && i != l && j != k && j != l

&& k != l )

printf ( "%d %d %d %d\n", i, j, k, l ) ;

}

}

}

}

}

150: Define an iterative function and a recursive function to print first ten terms of a Fibonacci series. Which of these two functions will run faster and why?

# include <stdio.h>

void ifibo ( int, int, int ) ;

void rfibo ( int, int, int, int ) ;

void main()

{

int first = 0, second = 1 ;

printf ( "First 10 terms of Fibonacci series are:\n" ) ;

ifibo ( first, second, 10 ) ;

printf ( "\n" ) ;

rfibo ( first, second, 0, 10 ) ;

}

void ifibo ( int old, int current, int terms )

{

int newterm, i ;

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

{

if ( i <= 1 )

newterm = i ;

else

{

newterm = old + current ;

old = current ;

current = newterm ;

}

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

}

}

void rfibo ( int old, int current, int count, int terms )

{

int newterm ;

if ( count == 0 || count == 1 )

{

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

count++ ;

terms-- ;

rfibo ( old, current, count, terms ) ;

}

else if ( terms >= 1 )

{

newterm = old + current ;

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

terms = terms - 1 ;

old = current ;

current = newterm ;

rfibo ( old, current, count, terms ) ;

}

}

Z