# KPIT TECHNOLOGIES

# WEEKLY REPORT

## WEEK 2- Report (DATE: 31/5/2024)

| Student name | Week | Branch | USN |
|---|---|---|---|
| Juweriah Herial | 2 | Circuit (ECE) | 1NH20EC053 |

## Yashavant Kanetkar Book 19th edition

**Question 50-100:**

**51) Write a program to print all prime numbers from 1 to 300.**

```
# include <stdio.h>
int main( )
{
int i, n = 1 ;
printf ( "\nPrime numbers between 1 and 300 are :\n1\t" ) ;
for ( n = 1 ; n <= 300 ; n++ )
{
i = 2 ;
for ( i = 2 ; i < n ; i++ )
{
if ( n % i == 0 )
break ;
}
if ( i == n )
printf ( "%d\t", n ) ;
}
return 0 ;
}
```

Output:

```
Prime numbers between 1 and 300 are:
1 2 3 5 7 11 13 17 19 23
29 31 37 41 43 47 53 59 61 67
71 73 79 83 89 97 101 103 107 109
113 127 131 137 139 149 151 157 163 167
173 179 181 191 193 197 199 211 223 227
229 233 239 241 251 257 263 269 271 277
281 283 293
```

**52) Write a program to add first seven terms of the following series using a for loop.**

```
 # include <stdio.h>
int main( )
{
int i = 1, j ;
float fact, sum = 0.0 ;
for ( i = 1 ; i <= 7 ; i++ )
{
fact = 1.0 ;
for ( j = 1 ; j <= i ; j++ )
fact = fact * j ;
sum = sum + i / fact ;
}
printf ( "Sum of series = %f\n", sum ) ;
return 0 ;
}
```

Output:

```
Sum of series = 2.718056
```

**53) Write a program to generate all combinations of 1, 2 and 3 using for loop.**

```
# include <stdio.h>
int main( )
{
int i = 1, j = 1, k = 1 ;
for ( i = 1 ; i <= 3 ; i++ )
{
for ( j = 1 ; j <= 3 ; j++ )
{
for ( k = 1 ; k <= 3 ; k++ )
printf ( "%d %d %d\n", i , j , k ) ;
}
}
return 0 ;
}
```

Output:

```
1 1 1
1 1 2
..
..
2 1 1
..
..
2 3 3
3 1 1
..
..
3 3 3
```

**54) Answer the following questions:**
**(a) The break statement is used to exit from:**
**1. An if statement**
**2. A for loop**
**3. A program**

**4. The main( ) function**
**(b) A do-while loop is useful when we want that the statements withinthe loop must be executed:**
**1. Only once**
**2. At least once**
**3. More than once**
**4. None of the above**
**(c) In what sequence the initialization, testing and execution of body is done in a do-while loop?**
**1. Initialization, execution of body, testing**
**2. Execution of body, initialization, testing**
**3. Initialization, testing, execution of body**
**4. None of the above**
**(d) Which of the following is not an infinite loop?**

```
1. int i = 1 ;
while ( 1 )
{
i++ ;
}
2. for ( ; ; ) ;
3. int t = 0, f ;
while ( t )
{
f = 1 ;
4. int y, x = 0 ;
do
{
y = x ;
```

**(e) Which of the following statements is true for the following program?**

```
# include <stdio.h>
int main( )
{
int x = 10, y = 100 % 90 ;
for ( i = 1 ; i <= 10 ; i++ ) ;
if ( x != y ) ;
printf ( "x = %d y = %d\n", x, y ) ;
return 0 ;
}
```

**1. The printf( ) function is called 10 times.**
**2. The program will produce the output x = 10 y = 10.**
**3. The ; after the if ( x != y ) will not produce an error.**
**4. The program will not produce any output.**
**5. The printf( ) function is called infinite times.**
**(f) Which of the following statement is true about a for loop used in a C program?**
**1. for loop works faster than a while loop.**
**2. All things that can be done using a for loop can also be done using a while loop.**
**3. for ( ; ; ) implements an infinite loop.**

**4. for loop can be used if we want statements in a loop to get executed at least once.**
**5. for loop works faster than a do-while loop.**

(a) 2. A for loop

(b) 2. At least once

(c) 1. Initialization, execution of body, testing

(d) 4. int y, x = 0; do { y = x; } while (x);

(e) 3. The ; after the if ( x != y ) will not produce an error.

(f) 2. All things that can be done using a for loop can also be done using a while loop.

**55) Attempt the following questions:**

**(a) Write a program to print the multiplication table of the number**

**entered by the user. The table should get displayed in the following**

**form:**

**29 * 1 = 29**

**29 * 2 = 58**

**…**

**(b) According to a study, the approximate level of intelligence of a**

**person can be calculated using the following formula:**

**i = 2 + ( y + 0.5 x )**

**Write a program that will produce a table of values of i, y and x, where y varies from 1 to 6, and, for each value of y, x varies from 5.5 to 12.5 in steps of 0.5.**

**(c) When interest compounds q times per year at an annual rate of r% for n years, the principal p compounds to an amount a as per the following formula**

**a=p(1+r/q) na**

**Write a program to read 10 sets of p, r, n & q and calculate the corresponding as.**

**(d) The natural logarithm can be approximated by the following series.**

**x-11 (**

**+**

**x-1**

**2**

**1(x-1**

**1**

**4**

**X**

**2**

**+**

**12**

**X**

**2**

**+**

**If x is input through the keyboard, write a program to calculate the sum of first seven terms of this series.**

**(e) Write a program to generate all Pythagorean Triplets with side length less than or equal to 30.**

**(f) Population of a town today is 100000. The population has increased steadily at the rate of 10% per year for last 10 years. Write a program to determine the population at the end of each year in the last decade.**

**(g) Ramanujan number (1729) is the smallest number that can be expressed as sum of two cubes in two different ways-1729 can be expressed as 13 + 123 and 93 + 103. Write a program to print all such numbers up to a reasonable limit.**

**(h) Write a program to print 24 hours of day with suitable suffixes like AM, PM, Noon and Midnight.**

**(i) Write a program to produce the following output:**

**1**

**2**

**3**

**4**

**5**

**6**

**7**

**8**

**9**

**10**

**a)**
```
#include <stdio.h>
```

```c
int main() {
    int num, i;
    printf("Enter a number: ");
    scanf("%d", &num);

    for (i = 1; i <= 10; i++) {
        printf("%d * %d = %d\n", num, i, num * i);
    }
return 0;
}
```

**b)**

```c
#include <stdio.h>
int main() {
    float x, y, i;
    for (y = 1; y <= 6; y++) {
        for (x = 5.5; x <= 12.5; x += 0.5) {
            i = 2 + (y + 0.5 * x);
            printf("y = %.1f, x = %.1f, i = %.2f\n", y, x, i);
        }
    }
    return 0;
}
```

**c)**

```c
#include <stdio.h>
#include <math.h>
int main() {
    float p,r,a;
    int n,q,i;
    for (i = 0; i < 10; i++) {
        printf("Enter p, r, n, q: ");
        scanf("%f %f %d %d", &p,&r,&n,&q);
        a=p*pow((1+r/(q*100)),n*q);
        printf("The amount is: %.2f\n", a);
    }
    return 0;
}
```

**d)**

```c
#include <stdio.h>
#include <math.h>
int main() {
    float x,term,sum;
    int i;
    printf("Enter x: ");
    scanf("%f",&x);
    sum=0;
    for (i=1;i<=7;i++) {
        term=pow((x-1),i)/i;
        sum+=(i%2==0?-1:1)*term;
    }
    printf("Sum of first seven terms: %f\n", sum);
    return 0;
}
```

**e)**

```
#include <stdio.h>
int main() {
    int a,b,c;
    for (a=1;a<=30;a++) {
        for (b=a;b<=30;b++) {
            for (c=b;c<=30;c++) {
                if (a*a+b*b==c*c) {
                    printf("(%d,%d,%d)\n",a,b,c);
                }
            }
        }
    }
    return 0;
}
```

**f)**

```
#include <stdio.h>
int main() {
    int year;
    double pop=100000;
    for (year=1;year<=10;year++) {
        pop*= 1.10;
        printf("Year %d: Population = %.0f\n", year, pop);
    }
    return 0;
}
```

**g)**

```
#include <stdio.h>
int isRamanujan(int num) {
    int a, b, c, d;
    for (a = 1; a * a * a < num; a++) {
        for (b = a + 1; a * a * a + b * b * b <= num; b++) {
            if (a * a * a + b * b * b == num) {
                for (c = a + 1; c * c * c < num; c++) {
                    for (d = c + 1; c * c * c + d * d * d <= num; d++) {
                        if (c * c * c + d * d * d == num && (a != c && b != d)) {
                            return 1;
                        }
                    }
                }
            }
        }
    }
    return 0;
}
int main() {
    int num;
    for (num = 1; num <= 10000; num++) {
        if (isRamanujan(num)) {
            printf("%d\n", num);
        }
    }
    return 0;
}
```

**h)**

```
#include <stdio.h>
int main() {
    int hour;
    for (hour = 0; hour < 24; hour++) {
        if (hour == 0) {
            printf("12 Midnight\n");
        } else if (hour == 12) {
            printf("12 Noon\n");
        } else if (hour < 12) {
            printf("%d AM\n", hour);
        } else {
            printf("%d PM\n", hour - 12);
        }
    }
    return 0;
}
```

**i)**

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

```
int main()
{
  int i, j, spc, rows=4, k, t = 1;
  spc = rows + 4 - 1;
  for (i = 1; i <= rows; i++) {
    for (k = spc; k >= 1; k--) {
      printf(" ");
    }

    for (j = 1; j <= i; j++) {
      printf("%d ", t++);
    }

    printf("\n");
    spc--;
  }
   return 0;
}
```

**56) Write a menu driven program which has following options:**

**1. Factorial of a number**

**2. Prime or not**

**3. Odd or even**

**4. Exit**

**Once a menu item is selected the appropriate action should be taken and once this action is finished, the menu should reappear. Unless the user selects the 'Exit' option the program should continue to work.**

```
# include <stdio.h>
# include <stdlib.h>
int main( )
{
int choice, num, i, fact ;
while ( 1 )
{
printf ( "\n1. Factorial\n" ) ;
printf ( "2. Prime\n" ) ;
printf ( "3. Odd / Even\n" ) ;
printf ( "4. Exit\n" ) ;
printf ( "Your choice? " ) ;
scanf ( "%d", &choice ) ;
switch ( choice )
{
case 1 :
printf ( "\nEnter number: " ) ;
scanf ( "%d", &num ) ;
fact = 1 ;
for ( i = 1 ; i <= num ; i++ )
fact = fact * i ;
printf ( "Factorial value = %d\n", fact ) ;
break ;
case 2 :
printf ( "\nEnter number: " ) ;
scanf ( "%d", &num ) ;
for ( i = 2 ; i < num ; i++ )
{
if ( num % i == 0 )
{
printf ( "Not a prime number\n" ) ;
break ;
}
}
if ( i == num )
printf ( "Prime number\n" ) ;
break ;
case 3 :
printf ( "\nEnter number: " ) ;
scanf ( "%d", &num ) ;
if ( num % 2 == 0 )
printf ( "Even number\n" ) ;
else
printf ( "Odd number\n" ) ;
break ;
case 4 :
exit ( 0 ) ; /* Terminates program execution */
default :
printf ( "Wrong choice!\a\n" ) ;
}
}
return 0 ;
}
```
Output:
```
1. Factorial
2. Prime
3. Odd / Even
4. Exit
Your choice?
1
Enter number: 5
Factorial value = 120
1. Factorial
2. Prime
3. Odd / Even
4. Exit
Your choice? 2
Enter number: 13
```

```
Prime number
1. Factorial
2. Prime
3. Odd / Even
4. Exit
Your choice? 3
Enter number: 13
Odd number
1. Factorial
2. Prime
3. Odd / Even
4. Exit
Your choice? 4
```

**57) What will be the output of the following programs?**

**(a)**
```
# include <stdio.h>

int main( )

{

char suite = 3 ;

switch ( suite )

{

case 1 :

printf ( "Diamond\n" ) ;

case 2 :

printf ( "Spade\n" ) ;

default :

printf ( "Heart\n" ) ;

}

printf ( "I thought one wears a suite\n" ) ;

return 0 ;

}
```

**(b)**
```
# include <stdio.h>

int main( )

{

int c = 3 ;

switch ( c )
```

```
{
case '3' :
printf ( "You never win the silver prize\n" ) ;
break ;
case 3 :
printf ( "You always lose the gold prize\n" ) ;
break ;
default :
printf ( "Of course provided you win a prize\n" ) ;
}
return 0 ;
}
(c) # include <stdio.h>
int main( )
{
int i = 3 ;
switch ( i )
{
case 0 :
printf ( "Customers are dicey\n" ) ;
case 1 + 2 :
printf ( "Markets are pricey\n" ) ;
case 4 / 2 :
printf ( "Investors are moody\n" ) ;
}
return 0 ;
}
(d) # include <stdio.h>
int main( )
{
```

```
int k ;
float j = 2.0 ;
switch ( k = j + 1 )
{
case 3 :
printf ( "Trapped\n" ) ;
break ;
default :
printf ( "Caught!\n" ) ;
}
return 0 ;
}
(e) # include <stdio.h>
int main( )
{
int ch = 'a' + 'b' ;
switch ( ch )
{
case 'a' :
case 'b' :
printf ( "Look at 10 ideas, 11th will occur to you\n" ) ;
case 'A' :
printf ( "If you have a good idea, project it\n" ) ;
case 'b' + 'a' :
printf ( "Have ideas, will fly\n" ) ;
}
return 0 ;
}
```

(a) Heart

I thought one wears a suite

(b) You always lose the gold prize

(c) Markets are pricey

Investors are moody

(d) Trapped

(e) Have ideas, will fly

**58) Point out the errors, if any, in the following programs:**

**(a)**
```
# include <stdio.h>
int main( )
{
int suite = 1 ;
switch ( suite ) ;
{
case 0 ;
printf ( "Club\n" ) ;
case 1 ;
printf ( "Diamond\n" ) ;
}
return 0 ;
}
```

**(b)**
```
# include <stdio.h>
int main( )
{
int temp ;
scanf ( "%d", &temp ) ;
switch ( temp )
{
case ( temp <= 20 ) :
printf ( "Oooooooohhhh! Damn cool!\n" ) ;
case ( temp > 20 && temp <= 30 ) :
printf ( "Rain rain here again!\n" ) ;
case ( temp > 30 && temp <= 40 ) :
printf ( "Wish I am on Everest\n" ) ;
default :
printf ( "Good old Nagpur weather\n" ) ;
}
return 0 ;
}
```

```
(c) # include <stdio.h>
int main( )
{
float a = 3.5 ;
switch ( a )
{
case 0.5 :
printf ( "The art of C\n" ) ; break ;
case 1.5 :
printf ( "The spirit of C\n" ) ; break ;
case 2.5 :
printf ( "See through C\n" ) ; break ;
}
return 0 ;
}
```

```
(d) # include <stdio.h>
int main( )
{
int a = 3, b = 4, c ;
c = b – a ;
switch ( c )
```

**{**

**case 1 || 2 :**

**printf ( "God give me a chance to change things\n" ) ;**

**break ;**

**case a || b :**

**printf ( "God give me a chance to run my show\n" ) ;**

**break ;**

**}**

**return 0 ;**

**}**

(a) The semicolon after the switch (suite) should not be there. It ends the switch statement prematurely. The case labels should not have semicolons; they should have colons.

(b) Case labels in switch statements must be constant expressions. You cannot use relational expressions directly in case labels.

(c)  The switch statement cannot handle floating-point numbers directly. Case labels must be integer constants.

(d) 1 || 2 is a logical OR operation which evaluates to 1. Thus, it is equivalent to case 1. a || b is also a logical OR operation that evaluates to 1 since a and b are non-zero. Thus, it is equivalent to case 1. The subtraction sign was incorrectly formatted as dash.

**59) Write a program to find the grace marks for a student using switch. The user should enter the class obtained by the student and the number of subjects he has failed in. Use the following logic:**

*** If the student gets first class and he fails in more than 3 subjects, he does not get any grace. Otherwise, he gets a grace of 5 marks per subject.**

*** If the student gets second class and he fails in more than 2 subjects, he does not get any grace. Otherwise, he gets a grace of 4 marks per subject.**

*** If the student gets third class and he fails in more than 1 subject, then he does not get any grace. Otherwise, he gets a grace of 5 marks.**

```c
#include <stdio.h>
int main() {
   int classObtained, subjectsFailed, graceMarks = 0;
   printf("Enter the class obtained by the student (1 for First Class, 2 for Second Class, 3 for Third Class): ");
   scanf("%d", &classObtained);
   printf("Enter the number of subjects the student has failed in: ");
   scanf("%d", &subjectsFailed);
   switch (classObtained) {
        case 1:
            if (subjectsFailed <= 3) {
                graceMarks = 5 * subjectsFailed;
            } else {
                graceMarks = 0;
            }
            break;
        case 2:
            if (subjectsFailed <= 2) {
                graceMarks = 4 * subjectsFailed;
            } else {
                graceMarks = 0;
            }
            break;
        case 3:
            if (subjectsFailed <= 1) {
                graceMarks = 5;
            } else {
                graceMarks = 0;
            }
            break;
        default:
            printf("Invalid class entered.\n");
            return 1;
    }
    printf("Grace marks awarded: %d\n", graceMarks);
    return 0;
}
```

**60) Write a function to calculate the factorial value of any integer entered through the keyboard.**

```
# include <stdio.h>

int fact ( int ) ;

int main( )

{

int num ;

int factorial ;

printf ( "\nEnter a number: " ) ;

scanf ( "%d", &num ) ;

factorial = fact ( num ) ;

printf ( "Factorial of %d = %ld\n", num, factorial ) ;

return 0 ;
}
int fact ( int num )
{
int i ;
int factorial = 1 ;
for ( i = 1 ; i <= num ; i++ )
factorial = factorial * i ;
return ( factorial ) ;
}
```

Output:

```
Enter a number: 6
Factorial of 6 = 720
```

**61) Write a function power ( a, b ), to calculate the value of a raised to b.**

```
# include <stdio.h>
float power ( float, int ) ;
int main( )
{
float x, pow ;
int y ;
printf ( "\nEnter two numbers: " ) ;
scanf ( "%f %d", &x, &y ) ;
pow = power ( x , y ) ;
printf ( "%f to the power %d = %f\n", x, y, pow ) ;
return 0 ;
}
float power ( float x, int y )
{
int i ;
float p = 1 ;
for ( i = 1 ; i <= y ; i++ )
p = p * x ;
return ( p ) ;
}
```

Output:

```
Enter two numbers: 1.5 3
1.500000 to the power 3 = 3.375000
```

**62) Define a function to convert any given year into its Roman equivalent. Use these Roman equivalents for decimal numbers: 1 – I, 5 – V, 10 – X, 50 – L, 100 – C, 500 – D, 1000 – M.**
**Example:**
**Roman equivalent of 1988 is mdccccIxxxviii.**
**Roman equivalent of 1525 is mdxxv.**

```
# include <stdio.h>
int romanise ( int, int, char ) ;
int main( )
{
int yr ;
printf ( "\nEnter year: " ) ;
scanf ( "%d", &yr ) ;
yr = romanise ( yr, 1000, 'm' ) ;
yr = romanise ( yr, 500, 'd' ) ;
yr = romanise ( yr, 100, 'c' ) ;
yr = romanise ( yr, 50, 'l' ) ;
yr = romanise ( yr, 10, 'x' ) ;
yr = romanise ( yr, 5, 'v' ) ;
romanise ( yr, 1, 'i' ) ;
return 0 ;
}
int romanise ( int y, int k, char ch )
{
int i, j ;
j = y / k ;
for ( i = 1 ; i <= j ; i++ )
printf ( "%c", ch ) ;
return ( y % k ) ;
}
```

Output:

```
Enter year: 1988
mdccccclxxxviii
```

**63)Point out the errors, if any, in the following programs:**

**(a)**

```
# include <stdio.h>
int addmult ( int, int )
int main( )
{
int i = 3, j = 4, k, l ;
k = addmult ( i, j ) ;
l = addmult ( i, j ) ;
printf ( "%d %d\n", k, l ) ;
return 0 ;
}
int addmult ( int x, int y )
{
int u, v ;
u = x + y ;
v = x * y ;
return ( u, v ) ;
}
```

**(b)**

```
# include <stdio.h>
int main( )
{
int a ;
a = message( ) ;
return 0 ;
}
void message( )
{
printf ( "Learn from him online at ykanetkar.com\n" ) ;
return ;
}
```

```
(c) # include <stdio.h>
int main( )
{
float a = 15.5 ;
char ch = 'C' ;
printit ( a, ch ) ;
return 0 ;
}
printit ( a, ch )
{
printf ( "%f %c\n", a, ch ) ;
}
```

```
(d) # include <stdio.h>
int main( )
{
let_us_c( )
{
printf ( "Learn C online…\n" ) ;
printf ( "At ykanetkar.com\n" ) ;
}
return 0 ;
}
```

(a) The function addmult should have a prototype declaration. The return (u, v); statement returns the value of v only, which is not the intended behavior.

(b) The message function is declared to return void, but it is used as if it returns an int.

(c) The printit function lacks a proper declaration and definition.

(d) The let_us_c function is used without a proper declaration and definition.

**64) State whether the following statements are True or False:**

**(a) Commonly used variables are available to all the functions in a program.**

**(b) To return the control back to the calling function we must use the keyword return.**

**(c) The same variable names can be used in different functions without any conflict.**

**(d) Every called function must contain a return statement.**

**(e) A function may contain more than one return statement.**

**(f) Each return statement in a function may return a different value.**

**(g) A function can still be useful even if you don't pass any arguments to it and the function doesn't return any value.**

**(h) Same names can be used for different functions without any conflict.**

**(i) A function may be called more than once from any other function.**

(a) False
(b) True
(c) True
(d) False
(e) True
(f) True
(g) True
(h) False
(i) True

**65)Answer the following questions:**

**(a) Any year is entered through the keyboard. Write a function to determine whether the year is a leap year or not.**

**(b) A positive integer is entered through the keyboard. Write a function to obtain the prime factors of this number.**

**For example, prime factors of 24 are 2, 2, 2 and 3, whereas prime factors of 35 are 5 and 7.**

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

#include<stdbool.h>
bool isLeapYear(int year) {
   if (year % 400 == 0) {
       return true ;
   } else if (year % 100 == 0) {
```

```c
        return false;
    } else if (year % 4 == 0) {
        return true;
    } else {
        return false;
    }
}
int main() {
    int year;
    printf("Enter a year: ");
    scanf("%d", &year);
    if (isLeapYear(year)==true) {
        printf("%d is a leap year.\n", year);
    } else {
        printf("%d is not a leap year.\n", year);
    }
    return 0;
}
```

(b)

```c
#include <stdio.h>
void primeFactors(int n) {
    printf("Prime factors of %d are: ", n);
    while (n % 2 == 0) {
        printf("%d ", 2);
        n = n / 2;
    }
    while (n % 3 == 0) {
        printf("%d ", 3);
        n = n / 3;
    }
    for (int i = 5; i <= n; i = i + 2) {
        while (n % i == 0) {
            printf("%d ", i);
            n = n / i;
        }
    }
    if (n > 3) {
        printf("%d", n);
    }
    printf("\n");
}

int main() {
    int n;
    printf("Enter a positive integer: ");
    scanf("%d", &n);

    if (n <= 0) {
        printf("Please enter a positive integer.\n");
    } else {
        primeFactors(n);
    }
    return 0;
}
```

**66) Write a function that receives 5 integers and returns the sum, average and standard deviation of these numbers. Call this function from main( ) and print the results in main( ).**

```
# include <stdio.h>

# include <math.h>

void stats ( int *, int *, double * ) ;

int main( )

{

int sum, avg ;

double stdev ;

stats ( &sum, &avg, &stdev ) ;

printf ( "Sum = %d \nAverage = %d \nStandard deviation = %lf\n",

sum, avg, stdev ) ;

return 0 ;

}

void stats ( int *sum, int *avg, double *stdev )

{

int n1, n2, n3, n4, n5 ;

printf ( "\nEnter 5 numbers: " ) ;

scanf ( "%d%d%d%d%d", &n1, &n2, &n3, &n4, &n5 ) ;

*sum = n1 + n2 + n3 + n4 + n5 ;

*avg = *sum / 5 ;

*stdev = sqrt ( ( pow ( ( n1 - *avg ), 2.0 ) + pow ( ( n2 - *avg ), 2.0 ) + \
pow ( ( n3 - *avg ), 2.0 ) + pow ( ( n4 - *avg ), 2.0 ) + \

pow ( ( n5 - *avg ), 2.0 ) ) / 4 ) ;

}
```

Output:

```
Enter 5 numbers: 10 20 30 40 50

Sum = 150

Average = 30

Standard deviation = 15.811388
```

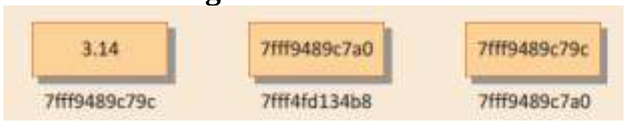
**67)Write a program that defines a function that calculates power of one number raised to another and factorial value of a number in one call.**

```
# include <stdio.h>

void power_fact ( float, int, int, float *, int * ) ;

int main( )

{

float a ;

int b, number, factorial ;

float pow ;

printf ( "Enter a and b for calculating a raised to b: " ) ;

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

printf ( "Enter number whose factorial is to be calculated: " ) ;

scanf ( "%d", &number ) ;

power_fact ( a, b, number, &pow, &factorial ) ;

printf ( "Power = %f Factorial = %d", pow, factorial ) ;

return 0 ;

}

void power_fact ( float x, int y, int num, float *power, int *fact )

{

float res = 1 ;

int i ;

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

res = res * x ;
*power = res ;
res = 1 ;
for ( i = 1 ; i <= num ; i++ )
res = res * i ;
*fact = res ;
}
```

Output:

```
Enter a and b for calculating a raised to b: 2 5
Enter number whose factorial is to be calculated: 6
Power = 32.000000 Factorial = 720
```

**68) Figure 9.4 shows three memory locations and values stored in them. Write a program to declare variables that implement the relationship shown. How will you print the values and addresses shown in the figure? On which machine the program should be executed to get such addresses?**



```
#include <stdio.h>
int main( )
{
float x = 3.14 ;
float *y ;
float **z ;
= &x ;
z = &y ;
printf ( "%p %p %p\n", &x, &y, &z ) ;
printf ( "%p %p\n", y, z ) ;
printf ( "%f %f %f", x, *y, **z ) ;
return 0 ;
}
```

Output:

```
0x7ffea600ed9c 0x7ffea600eda0 0x7ffea600eda8
0x7ffea600ed9c 0x7ffea600eda0
```

```
3.140000 3.140000 3.140000
```

**69)What will be the output of the following programs?**

**(a)**
```
# include <stdio.h>
void fun ( int, int ) ;
int main( )
{
int i = 5, j = 2 ;
fun ( i, j ) ;
printf ( "%d %d\n", i, j ) ;
return 0 ;
}
void fun ( int i, int j )
{
i = i * i ;
j = j * j ;
}
```

**(b)**
```
# include <stdio.h>
void fun ( int *, int * ) ;
int main( )
{
int i = 5, j = 2 ;
fun ( &i, &j ) ;
printf ( "%d %d\n", i, j ) ;
return 0 ;
}
void fun ( int *i, int *j )
{
*i = *i * *i ;
*j = *j * *j ;
}
```

**(c)** 
```
# include <stdio.h>
int main( )
{
float a = 13.5 ;
float *b, *c ;
b = &a ; /* suppose address of a is 1006 */
c = b ;
printf ( "%u %u %u\n", &a, b, c ) ;
printf ( "%f %f %f %f %f\n", a, *(&a), *&a, *b, *c ) ;
return 0 ;
}
```

(a) Output: 5 2

(b) Output: 25 4

(c) 1006 1006 1006

13.500000 13.500000 13.500000 13.500000 13.500000

**70) Point out the errors, if any, in the following programs:**

**(a)** 
```
# include <stdio.h>
int main( )
{
float p = 23.24 ;
int *q, **r ;
q = &p ;
r = &q ;
printf ( "%f %f\n", *q, **r ) ;
return 0 ;
}
```

**(b)** 
```
# include <stdio.h>
int main( )
{
char ch = 'A' ;
int k = 35 ;
float a = 3.14 ;
printf ( "%p %p %p\n", &ch, &k, &a ) ;
return 0 ;
}
```

**(c)**

```
# include <stdio.h>
void function ( int * ) ;
int main( )
{
int i = 35, *z ;
z = function ( &i ) ;
printf ( "%d\n", z ) ;
return 0 ;
}
void function ( int *m )
{
return ( *m + 2 ) ;
}
```

(a) In this program, q is a pointer to an integer, but it's assigned the address of a float variable p. Similarly, r is a pointer to a pointer to an integer, but it's assigned the address of a pointer to a float variable q. This results in a type mismatch error.

(b) No errors.

(c) Function Return Type Mismatch

**71)Attempt the following questions:**
**(a) Given three variables x, y, z, write a function to circularly shift their values to right. In other words, if x = 5, y = 8, z = 10, after circular shift y = 5, z = 8, x =10. Call the function with variables a, b, c to circularly shift values.**
**(b) Define a function that receives weight of a commodity in Kilograms and returns the equivalent weight in Grams, Tons and Pounds. Call this function from main( ) and print the results in main( ).**
**(c) Define a function to compute the distance between two points and use it to develop another function that will compute the area of the triangle whose vertices are A(x1, y1), B(x2, y2), and C(x3, y3). Use these functions to develop a function which returns a value 1 if the point (x, y) lines inside the triangle ABC, otherwise returns a value 0. Would you get any advantage if you develop these functions to work on call be reference principle?**

```
(a) void circular_shift(int *x, int *y, int *z) {
    int temp = *z;
    *z = *y;
    *y = *x;
    *x = temp;
}
```

```
b) #include <stdio.h>
void convert_weight(float kg, float *grams, float *tons, float *pounds) {
    *grams = kg * 1000;
    *tons = kg / 1000;
    *pounds = kg * 2.20462; // 1 kg = 2.20462 pounds
}
int main() {
    float kg = 5;
    float grams, tons, pounds;
    convert_weight(kg, &grams, &tons, &pounds);
    printf("Weight in grams: %.2f\n", grams);
    printf("Weight in tons: %.2f\n", tons);
    printf("Weight in pounds: %.2f\n", pounds);
    return 0;
}
```

**c)**

```
#include <stdio.h>
#include <math.h>
float distance(float x1, float y1, float x2, float y2) {
    return sqrt(pow(x2 - x1, 2) + pow(y2 - y1, 2));
}
float triangle_area(float x1, float y1, float x2, float y2, float x3, float y3) {
    float a = distance(x1, y1, x2, y2);
    float b = distance(x2, y2, x3, y3);
    float c = distance(x3, y3, x1, y1);
    float s = (a + b + c) / 2;
    return sqrt(s * (s - a) * (s - b) * (s - c));
}
int point_inside_triangle(float x, float y, float x1, float y1, float x2, float y2, float x3, float
y3) {
    float area_ABC = triangle_area(x1, y1, x2, y2, x3, y3);
    float area_ABP = triangle_area(x1, y1, x2, y2, x, y);
    float area_BCP = triangle_area(x2, y2, x3, y3, x, y);
```

```c
    float area_CAP = triangle_area(x3, y3, x1, y1, x, y);
    return (area_ABC == area_ABP + area_BCP + area_CAP);
}
int main() {
    float x1 = 0, y1 = 0;
    float x2 = 4, y2 = 0;
    float x3 = 2, y3 = 3;
    float x = 1, y = 1;
    if (point_inside_triangle(x, y, x1, y1, x2, y2, x3, y3))
        printf("Point (%.2f, %.2f) lies inside the triangle.\n", x, y);
    else
        printf("Point (%.2f, %.2f) does not lie inside the triangle.\n", x, y);
    return 0;
}
```

**72) We wish to calculate factorial value of an integer. As we know, factorial value of 4 is 4 * 3 * 2 * 1. This can also be expressed as 4! = 4 * 3! where '!' stands for factorial. Thus, factorial of a number can be expressed in the form of itself. Hence this logic can be programmed using recursion as shown in the following program.**

```c
# include <stdio.h>

int rec ( int ) ;

int main( )

{

int a, fact ;

printf ( "Enter any number: " ) ;

scanf ( "%d", &a ) ;

fact = rec ( a ) ;

printf ( "Factorial value = %d\n", fact ) ;

return 0 ;

}

int rec ( int x )

{

int f ;

if ( x == 1 )

return ( 1 ) ;

else

f = x * rec ( x - 1 ) ;
return ( f ) ;
}
```

Output:

```
Enter any number: 1
Factorial value = 1
Enter any number: 2
Factorial value = 2
Enter any number: 5
Factorial value = 120
```

**73)A 5-digit positive integer is entered through the keyboard, write a recursive function to calculate sum of digits of the 5-digit number.**

```
# include <stdio.h>
int rsum ( int ) ;
int main( )
{
int num, sum ;
int n ;
printf ( "Enter number: " ) ;
scanf ( "%d", &num ) ;
sum = rsum ( num ) ;
printf ( "Sum of digits is %d\n", sum ) ;
return 0 ;
}
int rsum ( int n )
{
int s, remainder ;
if ( n != 0 )
{
remainder = n % 10 ;
s = remainder + rsum ( n / 10 ) ;

}

else

return 0 ;

return s ;

}
```

Output:

```
Enter number: 12345

Sum of digits is 15
```

**74)A positive integer is entered through the keyboard, write a program to obtain the prime factors of the number. Modify the function suitably to obtain the prime factors recursively.**

```
# include <stdio.h>

void factorize ( int, int ) ;

int main( )

{

int num ;

printf ( "Enter a number: " ) ;

scanf ( "%d", &num ) ;

printf ( "Prime factors are: " ) ;

factorize ( num, 2 ) ;

return 0 ;

}

void factorize ( int n, int i )

{

if ( i <= n )

{

if ( n % i == 0 )

{

printf ( "%d ", i ) ;
n = n / i ;
}
else
i++ ;
factorize ( n, i ) ;
}
}
```

Output:

```
Enter a number: 60
Prime factors are: 2 2 3 5
```

**75 Write a recursive function to obtain the first 25 numbers of a Fibonacci sequence. In a Fibonacci sequence the sum of two successive terms gives the third term. Following are the first few terms of the Fibonacci sequence:**

**0 1 1 2 3 5 8 13 21 34 55 89….**

**Sol.**

```
#include<stdio.h>
int fibo ( int ) ;
int main( )
{
int terms = 25, i, n = 0 ;
for ( i = 1 ; i <= terms ; i++ )
{
printf ( "%d\t", fibo ( n ) ) ;
n++ ;
}
return 0 ;
}
int fibo ( int n )
{
if ( n == 0 || n == 1 )
return n ;
else
return ( fibo ( n - 1 ) + fibo ( n - 2 ) ) ;
}
```

Output:

```
0 1 1 2 3 5 8 13 21 34 55
89 144 233 377 610 987 1597 2584 4181 6765
10946 17711 28657 46368
```

**76)What will be the output of the following programs?**

**(a)**
```
# include <stdio.h>
int main( )
{
printf ( "I C, you C, we all C\n" ) ;
main( ) ;
return 0 ;
}
```

**(b)**
```
# include <stdio.h>
# include <stdlib.h>
int main( )
{
int i = 0 ;
i++ ;
if ( i <= 5 )
{
printf ( "C adds wings to your thoughts\n" ) ;
exit ( 0 ) ;
main( ) ;
}
return 0 ;
}
```

**Sol.** (a) I C, you C, we all C

I C, you C, we all C

I C, you C, we all C

...

(b) C adds wings to your thoughts

**77) Attempt the following questions:**

**(a) A positive integer is entered through the keyboard, write a function to find the binary equivalent of this number:**

**(1) Without using recursion**

**(2) Using recursion**

**(b) Write a recursive function to obtain the sum of first 25 natural numbers.**

**(c) There are three pegs labeled A, B and C. Four disks are placed on peg A. The bottom-most disk is largest, and disks go on decreasing in size with the topmost disk being smallest. The objective of the game is to move the disks from peg A to peg C, using peg B as an auxiliary peg. The rules of the game are as follows:**

**(1) Only one disk may be moved at a time, and it must be the top disk on one of the pegs.**

**(2) A larger disk should never be placed on the top of a smaller disk.**

**Write a program to print out the sequence in which the disks should be moved such that all disks on peg A are finally transferred to peg C.**

**a)**

```
#include <stdio.h>
void decimal_to_binary_without_recursion(int n) {
    int binary[32];
    int i = 0;
    while (n > 0) {
        binary[i] = n % 2;
        n = n / 2;
        i++;
    }
    printf("Binary equivalent: ");
    for (int j = i - 1; j >= 0; j--) {
        printf("%d", binary[j]);
    }
}
void decimal_to_binary_with_recursion(int n) {
    if (n > 1) {
        decimal_to_binary_with_recursion(n / 2);
    }
    printf("%d", n % 2);
}
int main() {
    int num;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    printf("Binary equivalent without recursion: ");
    decimal_to_binary_without_recursion(num);
    printf("\n");
    printf("Binary equivalent using recursion: ");
    decimal_to_binary_with_recursion(num);
    printf("\n");
    return 0;
}
```

**b)**

```
#include <stdio.h>
int sum_numbers(int n) {
    if (n == 0) {
        return 0;
    } else {
        return n + sum_numbers(n - 1);
    }
}
int main() {
    int n = 25;
    printf("Sum of first %d natural numbers: %d\n", n, sum_numbers(n));
    return 0;
}
```

**c)**

```
#include <stdio.h>
void move_disk(char from, char to) {
    printf("Move disk from peg %c to peg %c\n", from, to);
}
void tower_of_hanoi(int n, char source, char auxiliary, char destination) {
    if (n == 1) {
        move_disk(source, destination);
        return;
    }
    tower_of_hanoi(n - 1, source, destination, auxiliary);
    move_disk(source, destination);
    tower_of_hanoi(n - 1, auxiliary, source, destination);
}
```

```
int main() {
    int num_disks = 4;
    printf("Sequence of moves to transfer %d disks from peg A to peg C:\n", num_disks);
    tower_of_hanoi(num_disks, 'A', 'B', 'C');
    return 0;
}
```

**78) What will be the output of the following programs?**

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

```
int i = 0 ;
void val( ) ;
int main( )
{
printf ( "main's i = %d\n", i ) ;
i++ ;
val( ) ;
printf ( "main's i = %d\n", i ) ;
val( ) ;
return 0 ;
}
void val( )
{
i = 100 ;
printf ( "val's i = %d\n", i ) ;
i++ ;
}
```

**(b) # include <stdio.h>**

```
int main( )
{
static int count = 5 ;
printf ( "count = %d\n", count-- ) ;
```

```
if ( count != 0 )
main( ) ;
return 0 ;
}
(c) # include <stdio.h>
void fnc( ) ;
int main( )
{
func( ) ;
func( ) ;
return 0 ;
}
void func( )
{
auto int i = 0 ;
register int j = 0 ;
static int k = 0 ;
i++ ; j++ ; k++ ;
printf ( "%d % d %d\n", i, j, k ) ;
}
(d) # include <stdio.h>
int x = 10 ;
int main( )
{
int x = 20 ;
{
int x = 30 ;
printf ( "%d\n", x ) ;
}
printf ( "%d\n", x ) ;
```

```
return 0 ;
}
```

```
(a) main's i = 0
val's i = 100
main's i = 101
val's i = 100
(b)  count = 5
count = 4
count = 3
count = 2
count = 1
count = 0
(c)  1 1 1
1 1 1
(d)  30
20
```

**79) Point out the errors, if any, in the following programs:**

**(a)**
```
# include <stdio.h>
int main( )
{
long num = 2 ;
printf ( "%d\n", num ) ;
return 0 ;
}
```

**(b)**
```
# include <stdio.h>
int main( )
{
char ch = 200 ;
printf ( "%d\n", ch ) ;
return 0 ;
}
(c) # include <stdio.h>
int main( )
{
long float a = 25.345e454 ;
unsigned double b = 25 ;
printf ( "%lf %d\n", a, b ) ;
return 0 ;
}
(d) # include <stdio.h>
static int y ;
int main( )
{
static int z ;
printf ( "%d %d\n", y, z ) ;
return 0 ;
}
```

(a) Format Specifier Mismatch

(b)  Range of char

(c) Invalid Data Types

(d) No errors.

**80) State whether the following statements are True or False:**

**(a) The value of an automatic storage class variable persists between various function invocations.**

**(b) If the CPU registers are not available, the register storage class variables are treated as static storage class variables.**

**(c) If we try to use register storage class for a float variable the compiler will report an error message.**

**(d) The default value for automatic variable is zero.**

**(e) The life of static variable is till the control remains within the block in which it is defined.**

**(f) If a global variable is to be defined, then the extern keyword is necessary in its declaration.**

**(g) The address of register variable is not accessible.**

(a) False

(b) False

(c) True

(d) False

(e) False

(f) False

(g) True

**81) Write macro definitions for the following:**
**1. To test whether a character is a lowercase letter or not.**
**2. To test whether a character is an uppercase letter or not.**
**3. To test whether a character is an alphabet or not. Make use of the macros you defined in 1 and 2 above.**
**4. To obtain the bigger of two numbers.**

```
# include <stdio.h>

#define ISUPPER(x) ( x >= 65 && x <= 90 ? 1 : 0 )

#define ISLOWER(x) ( x >= 97 && x <= 122 ? 1 : 0 )

#define ISALPHA(x) ( ISUPPER(x) || ISLOWER(x) )

#define BIG(x,y) ( x > y ? x : y )

int main( )

{

char ch ;

int d, a, b ;

printf ( "\nEnter any alphabet/character: " ) ;

scanf ( "%c", &ch ) ;

if ( ISUPPER ( ch ) == 1 )

printf ( "You entered a capital letter\n" ) ;

if ( ISLOWER ( ch ) == 1 )

printf ( "You entered a small case letter\n" ) ;

if ( ISALPHA ( ch ) != 1 )
printf ( "You entered character other than an alphabet\n" ) ;

printf ( "Enter any two numbers: " ) ;

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

d = BIG ( a, b ) ;

printf ( "Bigger number is %d\n", d ) ;

return 0 ;

}
```

Output:

```
Enter any alphabet/character: A

You entered a capital letter

Enter any two numbers: 10 20

Bigger number is 20
```

**82)Write macro definitions with arguments for calculation of area and perimeter of a triangle, a square and a circle. Store these macro definitions in a file “areaperi.h”. Include this file in your program, and use the macro definitions for calculating area and perimeter for different squares, triangles and circles.**

```
#define PI 3.1415
#define PERIC( r ) ( 2 * PI * r )
#define AREAC( r ) ( PI * r * r )
#define PERIS( x ) ( 4 * x )
#define AREAS( x ) ( x * x )
#define PERIT( x, y, z ) ( x + y + z )
#define AREAT( b, h ) ( 0.5 * b * h )
# include <stdio.h>
# include "areaperi.h"
int main( )
{
int d, a, b ;
float sid1, sid2, sid3, sid, p_tri, p_cir, p_sqr, a_tri, a_cir,a_sqr ;
float r, base, height ;
printf ( "\nEnter radius of circle: " ) ;
scanf ( "%f", &r ) ;
p_cir = PERIC ( r ) ;
printf ( "Circumference of circle = %f\n", p_cir ) ;
a_cir = AREAC ( r ) ;
printf ( "Area of circle = %f\n", a_cir ) ;
printf ( "Enter side of a square: " ) ;
scanf ( "%f", &sid ) ;
p_sqr = PERIS ( sid ) ;
printf ( "Perimeter of square = %f\n", p_sqr ) ;
a_sqr = AREAS ( sid ) ;
printf ( "Area of square = %f\n", a_sqr ) ;
printf ( "Enter length of 3 sides of triangle: " ) ;
scanf ( "%f %f %f", &sid1, &sid2, &sid3 ) ;
p_tri = PERIT ( sid1, sid2, sid3 ) ;
printf ( "Perimeter of triangle = %f\n", p_tri ) ;
printf ( "Enter base and height of triangle: " ) ;
scanf ( "%f %f", &base, &height ) ;
a_tri = AREAT ( base, height ) ;
printf ( "Area of triangle = %f\n", a_tri ) ;
return 0 ;
}
```

Output:

```
Enter radius of circle: 5
Circumference of circle = 31.415001
Area of circle = 78.537498
Enter side of a square: 6
Perimeter of square = 24.000000
Area of square = 36.000000
Enter length of 3 sides of triangle: 3 4 5
Perimeter of triangle = 12.000000
Enter base and height of triangle: 4 6
Area of triangle = 12.000000
```

**83)Answer the following questions:**

**(a) A preprocessor directive is:**

**1. A message from compiler to the programmer**

**2. A message from compiler to the linker**

**3. A message from programmer to the preprocessor**

**4. A message from programmer to the microprocessor**

**(b) Which of the following are correctly formed #define statements?**

**#define INCH PER FEET 12**

**#define SQR (X) ( X * X )**

**#define SQR(X) X * X**

**#define SQR(X) ( X * X )**

**(c) State True or False:**

**1. A macro must always be written in capital letters.**

**2. A macro must always be accommodated in a single line.**

**3. After preprocessing when the program is sent for compilation**

**the macros are removed from the expanded source code.**

**4. Macros with arguments are not allowed.**

**5. In a macro call the control is passed to the macro.**

**(d) A header file is:**

**1. A file that contains standard library functions**

**2. A file that contains function declarations and macros**

**3. A file that contains user-defined functions**

**4. A file that is present in current working directory**

**(e) All macro substitutions in a program are done:**

**1. Before compilation of the program**

**2. After compilation of the program**

**3. During execution of the program**

**4. During linking of the program**

(a) A message from compiler to the programmer

(b) #define SQR(X) ( X * X )

(c)

1. False
2. False
3. True
4. False
5. True

(d) A file that contains function declarations and macros

(e) Before compilation of the program

**84) What will be the output of the following programs?**

**(a)**
```
# include <stdio.h>
int main( )
{
int i = 2 ;
# ifdef DEF
i *= i ;
# else
printf ( "%d\n", i ) ;
# endif
return 0 ;
}
```
**(b)**
```
# include <stdio.h>
# define PRODUCT(x) ( x * x )
```

```
int main( )
{
int i = 3, j, k, l ;
j = PRODUCT( i + 1 ) ;
k = PRODUCT( i++ ) ;
l = PRODUCT ( ++i ) ;
printf ( "%d %d %d %d\n", i, j, k, l ) ;
return 0 ;
}
```

**(c)**
```
# include <stdio.h>
# define PI 3.14
# define AREA( x, y, z ) ( PI * x * x + y * z ) ;
int main( )
{
float a = AREA ( 1, 5, 8 ) ;
float b = AREA ( AREA ( 1, 5, 8 ), 4, 5 ) ;
printf ( " a = %f\n", a ) ;
printf ( " b = %f\n", b ) ;
return 0 ;
}
```

(a) 2

(b) 6 16 9 36

(c) a = 59.140003

b = 2137.780029

85**)Attempt the following questions:**

**(a) If a macro is not getting expanded as per your expectation, how will you find out how is it being expanded by the preprocessor?**

**(b) Write macro definitions for the following:**

**1. To find arithmetic mean of two numbers.**

**2. To find absolute value of a number.**

**3. To convert an uppercase alphabet to lowercase.**

**4. To obtain the biggest of three numbers.**

**(c) Write macro definitions with arguments for calculation of Simple Interest and Amount. Store these macro definitions in a file "interest.h". Include this file in your program, and use the macro definitions for calculating Simple Interest and Amount.**

(a) gcc -E your_program.c

(b) 1. #define ARITHMETIC_MEAN(x, y) (((x) + (y)) / 2)

2. #define ABSOLUTE_VALUE(x) (((x) < 0) ? -(x) : (x))

3. #define TO_LOWER_CASE(x) (((x) >= 'A' && (x) <= 'Z') ? ((x) + 32) : (x))

4. #define MAX_OF_THREE(x, y, z) (((x) > (y) && (x) > (z)) ? (x) : ((y) > (z) ? (y) : (z)))

```
(c) // interest.h

#ifndef INTEREST_H

#define INTEREST_H

#define SIMPLE_INTEREST(principal, rate, time) ((principal * rate * time) / 100)

#define AMOUNT(principal, rate, time) (principal + SIMPLE_INTEREST(principal, rate,
time))

#endif

#include <stdio.h>

#include "interest.h"

int main() {

    float principal = 1000, rate = 5, time = 2;

    float interest = SIMPLE_INTEREST(principal, rate, time);

    float total_amount = AMOUNT(principal, rate, time);

    printf("Simple Interest: %.2f\n", interest);

    printf("Total Amount: %.2f\n", total_amount);

    return 0;

}
```

8**6)Write a program that interchanges elements at odd position with elements at even position in an array of 10 elements.**

```
# include <stdio.h>
int main( )
{
int num[ ] = { 12, 4, 5, 1, 9, 13, 11, 19, 54, 34 } ;
int i, t ;
for ( i = 0 ; i <= 9 ; i = i + 2 )
{
t = num[ i ] ;
num [ i ] = num [ i + 1 ] ;
num [ i + 1 ] = t ;
}
for ( i = 0 ; i <= 9 ; i++ )
printf ( "%d\t", num[ i ] ) ;
return 0 ;
}
```

Output:

```
4 12 1 5 13 9 19 11 34 54
```

**87) Write a program to copy the contents of a 5-element integer array into another array in reverse order.**

```
# include <stdio.h>
int main( )
{
int arr1[ 5 ], arr2[ 5 ], i, j ;
printf ( "\nEnter 5 elements of array:\n" ) ;
for ( i = 0 ; i <= 4 ; i++ )
scanf ( "%d", &arr1[ i ] ) ;
for ( i = 0, j = 4 ; i <= 4 ; i++, j-- )
arr2[ j ] = arr1[ i ] ;
printf ( "Elements in reverse order:\n" ) ;
for ( i = 0 ; i <= 4 ; i++ )
printf ( "%d\t", arr2[ i ] ) ;
return 0 ;
}
```

Output:

```
Enter 5 elements of array:
10 20 30 40 50
Elements in reverse order:
50 40 30 20 10
```

**88) An array contains 10 integers. Receive the number to be searched in the array as input. Write a program to search this number in the array and display the number of times it occurs in the array.**

```
# include <stdio.h>
int main( )
{
int num[ ] = { 7, 3, 5, 4, 6, 7, 2, 4, 6, 7 } ;
int n, i, count ;
printf ( "\nEnter an element to search: " ) ;
scanf ( "%d", &n ) ;
count = 0 ;
for ( i = 0 ; i <= 9 ; i++ )
{
if ( num[ i ] == n )
count++ ;
}
printf ( "Number %d is found %d time(s) in the array\n", n, count ) ;
return 0 ;
}
```

Output:

```
Enter an element to search: 7
Number 7 is found 3 time(s) in the array
```

**89) Answer the following questions:**

**(a) Are the following array declarations correct?**

**int a (25) ;**

**int size = 10, b[ size ] ;**

**(b) Which element of the array does this expression reference? num[ 4 ]**

**(c) What is the difference between the 5's in these two expressions?**

**int num[ 5 ] ;**

**num[ 5 ] = 11 ;**

**(d) What will happen if you try to put so many values into an array when you initialize it that the size of the array is exceeded?**

**(e) What will happen if you put too few elements in an array when you initialize it?**

**(f) What will happen if you assign a value to an element of an array whose subscript exceeds the size of the array?**

**(g) When you pass an array as an argument to a function, what actually gets passed?**

**(h) If you don't initialize a static array, what will its elements be set to?**

**(i) if int s[ 5 ] is a one-dimensional array of integers, how will you refer to the third element in the array using pointer notation?**

(a) Incorrect. Correct syntax: int a[25];

Correct

(b) It references the fifth element of the array num.

(c) The first expression declares an array of size 5, while the second tries to assign a value to the 6th element, which is out of bounds.

(d) It will result in a compilation error or warning.

(e) The remaining elements will be initialized to zero.

(f) It will result in undefined behavior.

(g) The address of the first element of the array.

(h) If declared at file scope, elements will be set to zero; if declared at block scope, elements will contain garbage values.

(i) *(s + 2) or s[2]

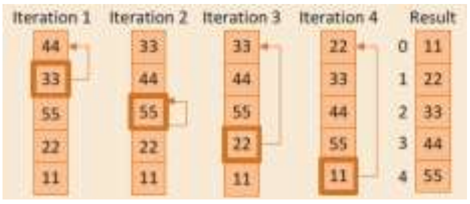
**90) Attempt the following questions:**

**(a) Twenty-five numbers are entered from the keyboard into an array. Write a program to find out how many of them are positive, how many are negative, how many are even and how many odd.**

**(b) If an array arr contains n elements, then write a program to check if arr[ 0 ] = arr[ n - 1 ], arr[ 1 ] = arr[ n - 2 ] and so on.**

**(c) Write a program using pointers to find the smallest number in an array of 25 integers.**

**(d) Implement the Insertion Sort algorithm shown in Figure 13.3 on a set of 25 numbers.**



**(e) Write a program which performs the following tasks:**
***Initialize an integer array of 10 elements in main( )**
***Pass the entire array to a function modify( )**

**In modify() multiply each element of array by 3 Return the control to main() and print the new array elements in main()**

**(f) For the following set of sample data, compute the standard deviation and the mean.**

**-6, -12, 8, 13, 11, 6, 7, 2, 6, 9, 10, 11, 10, 9, 2**

**The formula for standard deviation is**

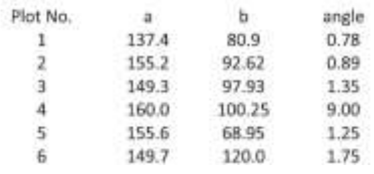
$$\frac{\sqrt{(x_i - \bar{x})^2}}{n}$$

**where x, is the data item and x is the mean..**

**(g) The area of a triangle can be computed by the sine law when 2 sides of the triangle and the angle between them are known.**

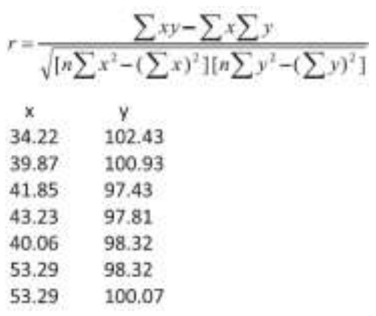
**Area (1/2) ab sin (angle)**

**Given the following 6 triangular pieces of land, write a program to find their area and determine which is largest.**



| Plot No. | a | b | angle |
|---|---|---|---|
| 1 | 137.4 | 80.9 | 0.78 |
| 2 | 155.2 | 92.62 | 0.89 |
| 3 | 149.3 | 97.93 | 1.35 |
| 4 | 160.0 | 100.25 | 9.00 |
| 5 | 155.6 | 68.95 | 1.25 |
| 6 | 149.7 | 120.0 | 1.75 |

**(h) For the following set of n data points (x, y), write a program to compute the correlation coefficient r, given by**



$$r = \frac{\sum xy - \sum x \sum y}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

| x | y |
|---|---|
| 34.22 | 102.43 |
| 39.87 | 100.93 |
| 41.85 | 97.43 |
| 43.23 | 97.81 |
| 40.06 | 98.32 |
| 53.29 | 98.32 |
| 53.29 | 100.07 |

**54.14 97.08**
**49.12 91.59**
**40.71 94.85**
**55.15 94.65**

**(i) The X and Y coordinates of 10 different points are entered through the keyboard. Write a program to find the distance of last point from the first point (sum of distances between consecutive points).**
**(j) A dequeue is an ordered set of elements in which elements may be inserted or retrieved from either end. Using an array simulate a dequeue of characters and the operations retrieve left, retrieve right, insert left, insert right. Exceptional conditions such as dequeue full or empty should be reported. Use two pointers left and right for this simulation.**

(a)

```
#include <stdio.h>
int main() {
    int arr[25];
    int positive_count = 0, negative_count = 0, even_count = 0, odd_count = 0;
    printf("Enter 25 numbers:\n");
    for (int i = 0; i < 25; i++) {
        scanf("%d", &arr[i]);
        if (arr[i] > 0)
            positive_count++;
        else if (arr[i] < 0)
            negative_count++;
        if (arr[i] % 2 == 0)
            even_count++;
        else
            odd_count++;
    }
    printf("Positive numbers: %d\n", positive_count);
    printf("Negative numbers: %d\n", negative_count);
    printf("Even numbers: %d\n", even_count);
    printf("Odd numbers: %d\n", odd_count);
    return 0;
}
```

(b)

```
#include <stdio.h>
int main() {
```

```c
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    int is_palindrome = 1;
    for (int i = 0; i < n / 2; i++) {
        if (arr[i] != arr[n - i - 1]) {
            is_palindrome = 0;
            break;
        }
    }
    if (is_palindrome) {
        printf("The array is a palindrome.\n");
    } else {
        printf("The array is not a palindrome.\n");
    }
    return 0;
}
```

(c)

```c
#include <stdio.h>
int main() {
    int arr[25];
    printf("Enter 25 integers:\n");
    for (int i = 0; i < 25; i++) {
        scanf("%d", &arr[i]);
    }
    int *ptr = arr;
    int min = *ptr;
    for (int i = 1; i < 25; i++) {
        if (*(ptr + i) < min) {
            min = *(ptr + i);
        }
    }
    printf("The smallest number is: %d\n", min);
    return 0;
}
```

(d) Iteration 1: The sorted sublist contains only the first element (22). The number 11 is removed from the unsorted sublist and inserted before 22 since 11 is smaller.
Iteration 2: The sorted sublist now contains the first two elements (11 and 22). The number 33 is removed from the unsorted sublist and inserted after 22 since 22 is smaller than 33 but larger than 11.

Iteration 3: The sorted sublist now contains the first three elements (11, 22, and 33). The number 44 is removed from the unsorted sublist and inserted after 33 since 33 is smaller than 44.
Iteration 4: The sorted sublist now contains all four elements (11, 22, 33, and 44). The number 55 is removed from the unsorted sublist and inserted at the end since it's the largest element so far.
(e)

```
#include <stdio.h>
void modify(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        arr[i] *= 3;
    }
}
int main() {
    int arr[10];
    printf("Enter 10 integers:\n");
    for (int i = 0; i < 10; i++) {
        scanf("%d", &arr[i]);
    }
    modify(arr, 10);
    printf("Modified array elements:\n");
    for (int i = 0; i < 10; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

(f)

```
#include <stdio.h>
#include <math.h>
int main() {
    int data[] = {-6, -12, 8, 13, 11, 6, 7, 2, 6, 9, 10, 11, 10, 9, 2};
    int n = sizeof(data) / sizeof(data[0]);
    double sum = 0;
    for (int i = 0; i < n; i++) {
        sum += data[i];
    }
    double mean = sum / n;
    double variance = 0;
    for (int i = 0; i < n; i++) {
        variance += pow(data[i] - mean, 2);
    }
    double std_dev = sqrt(variance / n);
    printf("Mean: %.2f\n", mean);
    printf("Standard Deviation: %.2f\n", std_dev);
    return 0;
}
```

(g)
```c
#include <stdio.h>
#include <math.h>
int main() {
    double sides[][3] = {
        {10,12,30},
        {8,15,40},
        {12,12,45},
        {9,10,60},
        {6,10,75},
        {15,18,90}
    };
    double max_area = -1;
    int max_index = -1;
    for (int i=0;i<6;i++) {
        double a =sides[i][0];
        double b =sides[i][1];
        double angle =sides[i][2];
        double area =0.5*a*b*sin(angle*M_PI / 180);
        if (area > max_area) {
            max_area =area;
            max_index =i+1;
        }
        printf("Area %d: %.2f\n", i + 1, area);
    }
    printf("Triangle %d has the largest area: %.2f\n", max_index, max_area);
    return 0;
}
```

(h)
```c
#include <stdio.h>
#include <math.h>
int main() {
    double data[][2] = {
        {54.14, 97.08},
        {49.12, 91.59},
        {40.71, 94.85},
        {55.15, 94.65}
    };
    int n = sizeof(data) / sizeof(data[0]);
    double sum_x = 0, sum_y = 0;
    for (int i = 0; i < n; i++) {
        sum_x += data[i][0];
        sum_y += data[i][1];
    }
    double mean_x = sum_x / n;
    double mean_y = sum_y / n;
    double numerator = 0, denominator_x = 0, denominator_y = 0;
    for (int i = 0; i < n; i++) {
        numerator += (data[i][0] - mean_x) * (data[i][1] - mean_y);
        denominator_x += pow(data[i][0] - mean_x, 2);
        denominator_y += pow(data[i][1] - mean_y, 2);
    }
    double r = numerator / sqrt(denominator_x * denominator_y);
    printf("Correlation Coefficient (r): %.2f\n", r);
    return 0;
}
```

(i)

```
#include <stdio.h>
#include <math.h>
float distance(int x1, int y1, int x2, int y2) {
    return sqrt(pow((x2 - x1), 2) + pow((y2 - y1), 2));
}
int main() {
    int x[10], y[10];
    printf("Enter the coordinates of 10 points:\n");
    for (int i = 0; i < 10; i++) {
        scanf("%d %d", &x[i], &y[i]);
    }
    float total_distance = 0;
    for (int i = 1; i < 10; i++) {
        total_distance += distance(x[i - 1], y[i - 1], x[i], y[i]);
    }
    printf("Total distance from the first point to the last point: %.2f\n", total_distance);
    return 0;
}
```

(j)

```
#include <stdio.h>
#include <stdbool.h>
#define MAX_SIZE 10
char dequeue[MAX_SIZE];
int left = -1, right = -1;
bool isEmpty() {
    return left == -1 && right == -1;
}
bool isFull() {
    return (right + 1) % MAX_SIZE == left;
}
void insertLeft(char value) {
    if (isFull()) {
        printf("Dequeue is full. Cannot insert.\n");
        return;
    }
    if (isEmpty()) {
        left = right = 0;
    } else {
        left = (left - 1 + MAX_SIZE) % MAX_SIZE;
    }
    dequeue[left] = value;
}
void insertRight(char value) {
    if (isFull()) {
        printf("Dequeue is full. Cannot insert.\n");
        return;
    }
    if (isEmpty()) {
        left = right = 0;
    } else {
        right = (right + 1) % MAX_SIZE;
    }
    dequeue[right] = value;
}
char retrieveLeft() {
    if (isEmpty()) {
        printf("Dequeue is empty. Cannot retrieve.\n");
        return '\0';
    }
    char value = dequeue[left];
    if (left == right) {
        left = right = -1;
    } else {
        left = (left + 1) % MAX_SIZE;
    }
    return value;
}
char retrieveRight() {
    if (isEmpty()) {
        printf("Dequeue is empty. Cannot retrieve.\n");
        return '\0';
    }
    char value = dequeue[right];
    if (left == right) {
        left = right = -1;
    } else {
        right = (right - 1 + MAX_SIZE) % MAX_SIZE;
    }
    return value;
}
int main() {
    insertRight('A');
    insertRight('B');
    insertLeft('C');
    printf("Retrieved from left: %c\n", retrieveLeft());
    printf("Retrieved from right: %c\n", retrieveRight());
    printf("Retrieved from right: %c\n", retrieveRight());
    printf("Retrieved from left: %c\n", retrieveLeft());
    printf("Retrieved from left: %c\n", retrieveLeft());
    return 0;
}
```

**91) Write a program to pick up the largest number from a 5 row by 5 column matrix.**

```
# include <stdio.h>
int main( )
{
int a[ 5 ][ 5 ] = {
{ 11, 1, 7, 9, 7 },
{ 13, 54, 56, 2, 5 },
{ 23, 43, 89, 22, 13 },
{ 14, 15, 17, 16, 19 },
{ 45, 3, 6, 8, 10 }
} ;
int i, j, big ;
big = a[ 0 ][ 0 ] ;
for ( i = 0 ; i <= 4 ; i++ )
{
for ( j = 0 ; j <= 4 ; j++ )
{
if ( a[ i ][ j ] > big )
big = a[ i ][ j ] ;
}
}
printf ( "\nLargest number in the matrix is %d\n", big ) ;
return 0 ;
}
```

Output:

```
Largest number in the matrix is 89
```

**92)Write a program to obtain transpose of a 3 x 5 matrix. The transpose of a matrix is obtained by exchanging the elements of each row with the elements of the corresponding column**

```
#include <stdio.h>
int main( )
{
int mat1[ 3 ][ 5 ] = {
1, 2, 3, 4, 5,
6, 7, 8, 9, 10,
11, 12, 13, 14, 15
} ;
int mat2[ 5 ][ 3 ] ;
int i, j ;
for ( i = 0 ; i < 3 ; i++ )
{
for ( j = 0 ; j < 5 ; j++ )
mat2[ j ][ i ] = mat1[ i ][ j ] ;
}
for ( i = 0 ; i < 5 ; i++ )
{
for ( j = 0 ; j < 3 ; j++ )
printf ( "%d\t", mat2[ i ][ j ] ) ;
printf ( "\n" ) ;
}
return 0 ;
}
```

Output:

```
1 6 11
2 7 12
3 8 13
4 9 14
5 10 15
```

**93)What will be the output of the following programs?**

**(a)**
```
# include <stdio.h>
int main( )
{
int n[ 3 ][ 3 ] = {
{ 2, 4, 3 }, { 6, 8, 5 }, { 3, 5, 1 }
} ;
printf ( "%d %d %d\n", *n, n[ 1 ][ 1 ], n[ 2 ][ 2 ] ) ;
return 0 ;
}
```

**(b)**
```
# include <stdio.h>
int main( )
{
int n[ 3 ][ 3 ] = {
{ 2, 4, 3 }, { 6, 8, 5 }, { 3, 5, 1 }
} ;
int i, *ptr ;
ptr = &n[ 0 ][ 0 ] ;
for ( i = 0 ; i <= 8 ; i++ )
printf ( "%d\n", *( ptr + i ) ) ;
return 0 ;
}
```

**(c)**
```
# include <stdio.h>
int main( )
{
int n[ 3 ][ 3 ] = {
2, 4, 3, 6, 8, 5, 3, 5, 1
} ;
int i, j ;
for ( i = 0 ; i <= 2 ; i++ )
for ( j = 0 ; j <= 2 ; j++ )
printf ( "%d %d\n", n[ i ][ j ], *( *( n + i ) + j ) ) ;
return 0 ;
}
```

(a) Output: 2 8 1

(b) 2

```
4
3
6
8
5
3
5
1
```

(c) 2 2

```
4 4
3 3
6 6
8 8
5 5
3 3
5 5
1 1
```

**94) Point out the errors, if any, in the following programs:**

**(a)**
```
# include <stdio.h>
int main( )
{
int twod[ ][ ] = {
2, 4, 6, 8
} ;
printf ( "%d\n", twod ) ;
return 0 ;
}
```

**(b) # include <stdio.h>**

**int main( )**

**{**

**int three[ 3 ][ ] = {**

**{ 2, 4, 3 }, { 6, 8, 2 }, { 2, 3, 1 }**

**} ;**

**printf ( "%d\n", three[ 1 ][ 1 ] ) ;**

**return 0 ;**

**}**

(a) Incomplete array dimension: In C, when initializing a 2D array without specifying both dimensions, you must provide the size of the second dimension. In this case, the size of the second dimension should be specified as 4.

(b) Incomplete array dimension: Similar to the previous error, when initializing a 2D array without specifying both dimensions, you must provide the size of the second dimension. In this case, the size of the second dimension should be specified based on the maximum number of elements in any row, which is 3.

**95) Attempt the following questions:**

**(a) How will you initialize a three-dimensional array threed[ 3 ][ 2 ][ 3]?**

**How will you refer the first and last element in this array?**

**(b) Match the following with reference to the program segment given**

**below:**

**int i, j, = 25 ;**

**int *pi, *pj = & j ;**

***pj = j + 5 ;**

**j = *pj + 5 ;**

**pj = pj ;**

***pi = i + j ;**

**Each integer quantity occupies 2 bytes of memory. The value assigned to i begins at (hexadecimal) address F9C and the value assigned to j begins at address F9E. Match the value represented by expression in left column with values in the right column.**

**1. &i a. 30**

**2. &j b. F9E**

**3. pj c. 35**

**4. *pj d. FA2**

**5. i e. F9C**

**6. pi f. 67**

**7. *pi g. unspecified**

**8. ( pi + 2 ) h. 65**

**9. (*pi + 2) i. F9E**

**10. * ( pi + 2 ) j. F9E**

**k. FAO**

**l. F9D**

**(c) Match the following with reference to the following program**

**segment:**

**int x[ 3 ][ 5 ] = {**

**{ 1, 2, 3, 4, 5 },**

**{ 6, 7, 8, 9, 10 },**

**{ 11, 12, 13, 14, 15 }**

**}, *n = &x ;**

**1. *( *( x + 2 ) + 1) a. 9**

**2. *( *x + 2 ) + 5 b. 13**

**3. *( *( x + 1) ) c. 4**

**4. *( *( x ) + 2 ) + 1 d. 3**

**5. * ( *( x + 1 ) + 3 ) e. 2**

**6. *n f. 12**

**7. *( n +2 ) g. 14**

**8. (*(n + 3 ) + 1 h. 7**

**9. *(n + 5)+1 i. 1**

**10. ++*n j. 8**

**k. 5**

**l. 10**

**m. 6**

**(d) Match the following with reference to the following program segment:**

**unsigned int arr[ 3 ][ 3 ] = {**

**{ 2, 4, 6 }, { 9, 1, 10 }, { 16, 64, 5 }**

**} ;**

**1. **arr a. 64**

**2. **arr < *( *arr + 2 ) b. 18**

**3. *( arr + 2 ) / ( *( *arr + 1 ) > **arr ) c. 6**

**4. *( arr[ 1 ] + 1 ) | arr[ 1 ][ 2 ] d. 3**

**5. *( arr[ 0 ] ) | *( arr[ 2 ] ) e. 0**

**6. arr[ 1 ][ 1 ] < arr[ 0 ][ 1 ] f. 16**

**7. arr[ 2 ][ [ 1 ] & arr[ 2 ][ 0 ] g. 1**

**8. arr[ 2 ][ 2 ] | arr[ 0 ][ 1 ] h. 11**

**9. arr[ 0 ][ 1 ] ^ arr[ 0 ][ 2 ] i. 20**

**10. ++**arr + --arr[ 1 ][ 1 ] j. 2**

**k. 5**

**l. 4**

**(e) Write a program to find if a square matrix is symmetric.**

**(f) Write a program to add two 6 x 6 matrices.**

**(g) Write a program to multiply any two 3 x 3 matrices.**

**(h) Given an array p[ 5 ], write a function to shift it circularly left by two positions. Thus, if the original array is { 15, 30, 28, 19, 61 } then after shifting it will be { 28, 19, 61, 15, 30 } Call this function for a 4 x 5 matrix and get its rows left shifted.**

(a)
```
int threed[3][2][3] = {
    {
        {1, 2, 3},
        {4, 5, 6}
    },
    {
        {7, 8, 9},
        {10, 11, 12}
    },
    {
        {13, 14, 15},
        {16, 17, 18}
    }
};
```

(b)

1. &i - e. F9C
2. &j - l. F9E
3. pj - h. 65
4. *pj - k. FA0
5. i - a. 30
6. pi - f. 67
7. *pi - g. unspecified
8. (pi + 2) - j. F9E
9. (*pi + 2) - c. 35
10. *(pi + 2) - i. F9D

(c)

1. *( *(x + 2) + 1) - g. 14
2. *( *x + 2 ) + 5 - b. 18
3. *( *(x + 1) ) - e. 2
4. *( *(x) + 2 ) + 1 - d. 3
5. ( *(x + 1) + 3 ) - a. 9
6. *n - f. 12
7. *(n +2 ) - j. 8
8. (*(n + 3 ) + 1 - l. 10
9. *(n + 5)+1 - i. 1
10. ++*n - c. 4

(d)

1. **arr - f. 16

2. **arr < *( *arr + 2 ) - b. 18
3. *( arr + 2 ) / ( *( *arr + 1 ) > **arr ) - a. 64
4. *( arr[ 1 ] + 1 ) | arr[ 1 ][ 2 ] - d. 3
5. *( arr[ 0 ] ) | *( arr[ 2 ] ) - e. 0
6. arr[ 1 ][ 1 ] < arr[ 0 ][ 1 ] - i. 20
7. arr[ 2 ][ [ 1 ] & arr[ 2 ][ 0 ] - h. 11
8. arr[ 2 ][ 2 ] | arr[ 0 ][ 1 ] - g. 1
9. arr[ 0 ][ 1 ] ^ arr[ 0 ][ 2 ] - k. 5
10. ++**arr + --arr[ 1 ][ 1 ] - j. 2

(e)
```
#include <stdio.h>
#define SIZE 3
int isSymmetric(int mat[SIZE][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (mat[i][j] != mat[j][i]) {
                return 0;
            }
        }
    }
    return 1;
}
int main() {
    int mat[SIZE][SIZE] = {
        {1, 2, 3},
        {2, 4, 5},
        {3, 5, 6}
    };
    if (isSymmetric(mat)) {
        printf("The matrix is symmetric.\n");
    } else {
        printf("The matrix is not symmetric.\n");
    }
    return 0;
}
```

(f)

```
#include <stdio.h>
#define ROWS 6
#define COLS 6
void addMatrices(int mat1[][COLS], int mat2[][COLS], int result[][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}
void printMatrix(int mat[][COLS]) {
    for (int i = 0; i < ROWS; i++) {
        for (int j = 0; j < COLS; j++) {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}
int main() {
    int mat1[ROWS][COLS] = { /* fill with values */ };
    int mat2[ROWS][COLS] = { /* fill with values */ };
    int result[ROWS][COLS];
    addMatrices(mat1, mat2, result);
    printf("Resultant matrix:\n");
    printMatrix(result);
    return 0;
}
(g)  #include <stdio.h>
#define SIZE 3
void multiplyMatrices(int mat1[][SIZE], int mat2[][SIZE], int result[][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            result[i][j] = 0;
            for (int k = 0; k < SIZE; k++) {
                result[i][j] += mat1[i][k] * mat2[k][j];
            }
        }
    }
}
void printMatrix(int mat[][SIZE]) {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}
int main() {
    int mat1[SIZE][SIZE] = { /* fill with values */ };
    int mat2[SIZE][SIZE] = { /* fill with values */ };
    int result[SIZE][SIZE];
    multiplyMatrices(mat1, mat2, result);
    printf("Resultant matrix:\n");
    printMatrix(result);
    return 0;
}
```

```
(h) #include <stdio.h>

#define SIZE 5

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

    int temp[SIZE];

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

        temp[(i + SIZE - 2) % SIZE] = arr[i];

    }

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

        arr[i] = temp[i];

    }

}

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

    for (int i = 0; i < n; i
```

9**6) 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".**

```
# 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 ;
}
```

Output:

```
Enter a string: Nagpur
Enter position and no. of characters to extract: 3 10
The substring is: pur
```

**97)Write a program that converts a string like "124" to an integer 124.**

```
# 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 ;

}
```

Output:

```
Enter a string containing a number: 237

The number is: 237
```

**98)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 ;
}
```
Output:
```
BA
BAB
BABBA
BABBABAB
BABBABABBABBA
```

**99)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 ;
}
```
(a)
```
A
A
```
(b)
```
Get organized! Learn C!!
Get organized! Learn C!!
Get organized! Learn C!!
t
```
(c)
```
B B
o o
r r
r r
```

o o

w w

e e

r r

s s

o

f f

f

b b

o o

o o

k k

s s

s

p p

o o

i i

l l

l

t t

h h

e e

e

s s

y y

m m

m m

e e

t t

r r

```
y y
  y
o o
f f
  f
s s
h h
e e
l l
v v
e e
s s
```

(d) Hello

Hello

(e) Morning f

e

(f) 1 2 4

**100) Fill in the blanks:**
**(a) "A" is a ____________ whereas 'A' is a _____________.**
**(b) A string is terminated by a ______ character.**
**(c) The array char name[ 10 ] can consist of a maximum of ______ characters.**
**(d) The array elements are always stored in __________ memory locations.**
**Sol.** (a) String literal
(b) null
(c) 9
(d) contiguous