PPS PYQ

Q. Explain the difference b/w call by reference and call by value with the help of suitable example.

Sol:

Call by reference

- Calling function send address of data
- The formal parameters are pointer variables.
- It is more efficient
- It require less memory
- original value get affected or value is modified.

```
void main()
{
    int a, b, sum;
    int add (int*, int*);
    printf("enter first value:");
    scanf("%d", &a);
    printf("enter second value:");
    scanf("%d", &b);
    sum = add (&a, &b);
}
int add (int* x, int* y)
{
    int z;
    z = *x + *y
    printf("addition = %d", z);
    return z;
```

Call by value

- Calling function send copies to data.
- The formal parameters are ordinary variables.
- It is less efficient
- It require more memory.
- No effect on original parameter after modifying.

```
void main()
{
    int a, b, sum;
    int add(int, int);
    printf("enter first value:");
    scanf("%d", &a);
    printf("enter second value:");
    scanf("%d", &b);
    sum = add(a,b);
}
int add(int x, int y)
{
    int z;
    z = x+y;
printf("addition = %d", z);
    return z;
}
```

Q. Fibonacci series

Sol:

Fibonacci series:
Fibonacci series is a function series.

$$F_n = F_{n-1} + F_{n-2}$$

Using Recursion

```
#include <stdio.h>
int fib(int n)
{
if (n<=1)
    return n;
return fib(n-1) + fib(n-2);
}
int main()
{
int n = 20;
printf("%d", fib(n));
getchar();
return 0;
}
```

Q. Write the differences structure and union. Compare them with the help of an example.

Ans:

Structure

→ A structure is a procedure capable of storing data belonging to different types.

→ All the members can be assigned values at time.

→ Value assigned to one member will not cause the change in other members.

Declaration

```
struct <tag>
{
  datatype member 1;
  datatype member 2;
};
```

Example

```
struct s
char c;
int x;
float y;
} s;
```

```
#include <stdio.h>
#include <string.h>
struct pokemon {
    int hp;
    int attack;
    int speed;
};
int main() {
    struct pokemon pikachu;
    pikachu.hp = 60;
    pikachu.attack = 70;
    pikachu.speed = 100;

    printf("%d\n", pikachu.hp);
    printf("%d\n", pikachu.attack);
    printf("%d\n", pikachu.speed);

    return 0;
}
```

OUTPUT

60
70
100

UNION

```
#include <stdio.h>
#include <string.h>
typedef union pokemon {
    int hp;
    int attack;
    int speed;
} pokemon;

int main() {
    pokemon pikachu;
    pikachu.hp = 60;
    pikachu.attack = 70;
    pikachu.speed = 100;

    printf("%d\n", pikachu.hp);
    printf("%d\n", pikachu.attack);
    printf("%d\n", pikachu.speed);

    return 0;
}
```

OUTPUT

```
1634429264
1634429264
1634429264
```

Union

- Union is a derived data type or element holding objects of different properties (data type, size, etc).
- Only one member can be assigned a value at a time.
- Value assigned to one member may cause the change in value of other members.

Declaration

```
union <tag>
{
datatype member 1;
datatype member 2;
};
```

Example:

```
union tag
{
char c;
int x;
float y;
};
```

Q. What is an array? Write a C program to find the average of 10 numbers of an array.

Sol (7)

1) An array is a data structure that contains a group of elements.
2) An array is a variable that stores multiple values of the same type in a single location in memory.
3) Array are commonly used in computer programs to organize data so that a related set of values can be easily sorted or searched.

Syntax - type array Name [array size];
Example - double balance [10];

Program:

```
#include <stdio.h>
int main() {
int array[10] = {1,2,3,4,5,6,7,8,9,0};
int sum, loop;
float avg;
sum = avg = 0;
for (loop = 0; loop < 10; loop++) {
sum = sum + array[loop];
}
avg = (float) sum / loop
printf ("Average of array is %f", avg);
return 0;
}
```

OUTPUT

Average of array value is 4.50.

Q. Which

Sol

Even

Step1:
Step2:
Step3:
Step4:
Step5:
Step6:
Step7:
Step8:

Q. Which of the sorting algorithm is better than others and why? Explain

Sol

Even though quick-sort has a worst case run time of $O(n^2)$, quicksort is considered the best sorting because it is very efficient on the average. Quicksort is generally considered the "fastest" sorting algorithm. The best sorting algorithm depends on the situation, but here are some of the most efficient sorting algorithm. Quicksort is one of the most efficient sorting algorithms.

Algorithm for quick sort:

Step1: Choose the highest index value has pivot.

Step2: Take two variable to point left and right of the list excluding pivot

Step3: left points to the low index

Step4: right point to the high

Step5: while value at left is less than pivot move right.

Step6: while value at right is greater than pivot move left.

Step7: if both step 5 and step does not match swap left an right.

Step8: if left & right, the point where they met is new pivot.

Q. What do you mean by call by reference Explain with the help of a function which swaps two numbers.

Ans 29

Call by reference

- Calling function send address of data.
- The formal parameters are pointer variables.
- It is more efficient.
- It require less memory.
- Original value get affected or value is modified.

Program:

```
#include <stdio.h>
void swap (int x, int y);
int main() {
int a=10, b=20;
swap (a,b);
printf ("a = %d b = %d\n", a,b);
return 0;
}
void swap (int x, int y) {
int t;
t = x;
x = y;
y = t;
printf (" x = %d y = %d\n", x,y);
}
```

OUTPUT

x = 20 y = 10 a = 10 b = 20.

y. reference
function.

of data.
pointer

r value

Q. Reverse a string without using built-in

Soln

```
#include <stdio.h>
#include <string.h>
int main() {
char str[40];
puts("enter a string");
scanf("%.[^\n]s", str);
// size
int size = 0;
int k = 0;
while (str[k] != '\0') {
    size++;
    k++;
}
for (int i=0, j=size-1; i<=j; i++, j--) {
char temp = str[i];
str[i] = str[j];
str[j] = temp;
}
puts("The reverse string is:");
puts(str);
return 0;
}
```

OUTPUT

```
Enter a string
Dinesh
The reverse string is:
hsenid
```

Q. Write a 'recursive' c program to print the following:

(a) GCD of two numbers.

Ans

```
#include <stdio.h>
int main () {
int n1, n2;
printf ("\n Enter two number: \n");
scanf ("%d\n%d", &n1, &n2);

while (n1 != n2) {
if (n1 > n2) {
    n1 = n1 - n2;
}
  else {
    n2 = n2 - n1;
  }
}
printf ("\n GCD of two number is:");
return 0;
}
```

OUTPUT

```
Enter two number:
10
5
GCD of two number is: 5
```

gram to print

per:\n");
%N2);

Q. Write the program to swap value of two variables using call by reference.

Soln

```
#include <stdio.h>
void swap (int*, int*);
int main () {
int a = 10;
int b = 20;
Printf (" Before swapping the values in
main a = %d, b = %d \n", a, b);

swap (&a, &b);
printf (" After swapping values in main
a = %d, b = %d \n", a, b);
}

    void swap (int* a, int* b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
printf (" After swapping value in function
a = %d, b = %d \n", *a, *b);
}
```

OUTPUT

Before swapping the values in main a = 10,
b = 20.

After swapping values in function a = 20,
b = 10

After swapping values in main a = 20,
b = 10

2. Write a program to accept a string as input from user and determine its length. [Do not use built in length function strlen().]

Sol:

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[1000];
    int i;
    printf("Enter the string:");
    scanf("%s", str);
    for (i=0; str[i]!='\0'; ++i);
    printf("Length of str is:%d", i);
    return 0;
}
```

Output

```
Enter the string
Dinkar
Length of str is:6.
```

Q. Define calling and called functions with suitable examples. What do you mean by call by value and call by reference? Write C programs for swapping of two number using call by value and call by reference techniques.

Ans:

Function calling: A function call is an important part of C programming language. It is called inside a program whenever it is required to call a function. It is only called by its name in the main() function of a program.

Call by value

- Calling function send copies to data.
- The formal parameters are ordinary variables.
- It is less efficient
- It require more memory.
- No effect on original parameter after modifying.

Program:

```
#include <stdio.h>
#include <conio.h>
void swap(int a, int b);
void main()
{
int a,b;
printf("Enter two numbers");
scanf("%d%d", &a, &b);
swap(a,b);
printf("a=%d b=%d", a,b);
getch();
}
void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
printf("a=%d b=%d", a,b);
}
```

OUTPUT

```
Enter two numbers
2
3
a = 3  b = 2
a = 2  b = 3
```

Call by reference

- Calling function send address of data
- The formal parameters are pointer variables.
- It is more efficient.
- It require less memory.
- Original value gets affected if value is modified.

Program:

```
#include<stdio.h>
#include<conio.h>
void swap(int*, int*);
int main(){
  int a,b;
printf("Enter two numbers");
scanf("a=%d b=%d", a,b);
getch();
}
  void swap(int* a, int* b);
   int temp;
   temp = *a;
  *a = *b;
  *b = temp;
printf("a = %d b = %d", a,b);
}
```

OUTPUT

Enter two number.

2

3

a = 3 b = 2

a = 2 b = 3

Q. What is recursive function? Explain. For given integer n, write a program in C to find the sum of the following series using recursion:

$S = 1 + 2 + 3 + \dots + n$.

Ans/

A recursive function is a function that calls itself. In programming, a recursive function is made up of two parts: the base case and the recursive case.

OR

A recursive function is a function that repeats or uses its own previous term to calculate subsequent terms and thus forms a sequence of terms.

Program:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  clrscr();
  void series(int);
  int n;
  printf("Number:");
  scanf("%d", &n);
  series(n);
  getch();
}
```

```
void series (int n)
{
static int i = 1;
printf ("%d \n", i);
if (i >= n)
    return;
else
    i++;
    series (n);
}
```

OUTPUT

Number : 10

Result

1
2
3
4
5
6
7
8
9
10

Q. Write a program in C to print the sum of the following series:

2 + 4 + 6 + 8 + 10 .....

Sol:

```
#include <stdio.h>
int main()
{
int n, sum = 0, i = 0;
printf("Enter the range of number:");
scanf("%d", &n);
while (i <= n)
{
sum = i;
i += 2;
}
printf("The Sum of the series = %d", sum);
}
```

Output

Enter the range of number: 9

The sum of the series = 20.

Q. Write a C program to find result matrix after multiplying two given matrices using 2-D arrays.

Soln

```
#include<stdio.h>
int main()
{
// 1st matrix order
int m;
printf("Enter no. of row of 1st matrix:");
scanf("%d", &m);
int n;
printf("Enter no. of column of 1st matrix:");
scanf("%d", &n);
int a[m][n];
// input and print the first matrix.
printf("\n Enter element of 1st matrix:");
for(int i=0; i<m; i++)
{
for(int j=0; j<n; j++)
scanf("%d", &a[i][j]);
}
// 2nd matrix order
int p;
printf("Enter no. of row of 2nd matrix:");
scanf("%d", &p);
int q;
printf("Enter no. of column of 2nd matrix:");
scanf("%d", &q);
int b[p][q];
```

// input the second matrix
printf("\n Enter elements of 2nd matrix:");
for(int i=0; i<r; i++)
{
for(int j=0; j<q; j++)
scanf("%d", &b[i][j]);
}

// check
if(c!=p)
printf("The matrices cannot be
multiplied");
}
else
{
// multiplication
int sum[m][q];
for(int i=0; i<m; i++)
{
for(int j=0; j<m; j++)
sum[i][j]=0;
// i now row of a, j column of b
for(int k=0; k<m; k++)
sum[i][j] += a[i][k] * b[k][j];

Print
like
Printf("The resultant matrix is\n");
for(int i=0; i<m; i++)
{
for(int j=0; j<q; j++)
printf("%d", sum[i][j]);
}
printf("\n");
}
}
return 0;
}

OUTPUT

Enter no. of rows of 1st matrix : 2
" " " columns " " " : 2

Enter element of 1st matrix :
1 0
0 1

Enter no. of rows of 2nd matrix : 2
Enter no. of column of 2nd matrix : 2

Enter elements of 2nd matrix :
1 0
0 1

RESULT

The resultant matrix is :
1 0
0 1

Thought Question

Q. Write a C program that converts a string like "123" to an integer 123.

Ans

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char str[100];
    int number;
    printf("Enter a string:");
    scanf("%s", str);
    number = atoi(str);
    printf("Converted integer: %d\n", number);
    return 0;
}
```

Q. With the help of an example, differentiate b/w static and dynamic memory allocation.

Ans:

There are three types of allocation:
(i) Static (ii) Dynamic (iii) Automatic

Static memory allocation:

- Variable get allocated permanently.
- Static memory allocation use stack as a data structure
- It is less efficient than dynamic memory allocation
- There is no memory reusability.

Example :- int a[10];

Dynamic memory allocation:

- Variables get allocated only if the program unit gets active.
- Allocation is done during program execution
- More efficient than static allocation.
- Memory reusability is possible.

Example :- a = malloc (sizeof(int) * 10);

Q. Factorial of a given number.

Sol:-

```
#include<stdio.h>
int main()
{
int n;
printf("Enter a number");
scanf("%d", &n);
// 5! = 1*2*3*4*5
int product = 1;
for(int i=1; i<=n; i++){
product = product * i;
}
printf("The factorial is:%d", product);
return 0;
}
```

Output

Enter a number: 5

Result

The factorial is: 120.

Q. What is type casting? Explain its classification through examples.

Sol/n

Type casting refers to changing an variable of one data type into another. The compiler will automatically change one type of data into another if it makes sense.

Type conversion in C can be classified into the following two types.

1. Implicit Type conversion:

→ When the type conversion is performed automatically by the compiler without programmers intervention, such type of conversion is known as implicit type conversion or type promotion.

Example :-

```
int x;
for(x=97; x<=122; x++){
printf("%c", x);
}
```

→ Implicit casting from int to char with use of %c formate specifier.

2. Explicit Type conversion

→ The type conversion performed by the programmer by posing the data type of the expression of specific type is known as explicit type conversion.

Example:

```
int x;
for (x=97; x<122; x++) {
printf("%c", (char) x);
```

→ here Explicit casting from int to char

Q. Explain the following bitwise operators:
(a) Bitwise AND
(b) Bitwise OR
(c) Bitwise XOR
(d) Bitwise left Shift

Sol:
In arithmetic - logic unit, mathematical operations like: addition, subtraction, multiplication and division are done in bit - level.

(a) Bitwise AND operator &
→ The output of bitwise AND is 1 if the corresponding bits of two operands is 1.

Let suppose the bitwise AND operation of two integers 12 and 25.

12 = 00001100 (in binary)
25 = 00011001 (in binary)

Bit wise AND operation of 12 and 25.

```
  00001100
& 00011001
  --------
  00001000   = 8 (in Decimal).
```

(b) Bitwise OR operator

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. bitwise OR operator is denoted by |.

12 = 00001100 (in Binary)
25 = 00011001 (" ")

Bitwise OR operation of 12 and 25.

```
  00001100
| 00011001
  --------
  00011101   = 29 (in Decimal).
```

(c) Bitwise XOR operator

→ The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

12 = 00001100 (in Binary)
25 = 00011001 (" ")

Bitwise XOR operation of 12 and 25

```
  00001100
^ 00011001
  --------
  00010101  = 21 (in Decimal)
```

(d) Bitwise left shift operator.

→ Left shift operator shifts all bits towards left by a certain number of specified bits. The symbol of the left shift operator is <<.

212 = 11010100 (in binary)
212 << 1 = 110101000 (" ")
212 << 0 = 11010100 (shift by 0)

NOTE: —

&, |, <<, ^ are symbols of bitwise operators.

2. Define enumerated data types with suitable example.

5 Marks

The enumerated data types is also known as the enium in the c language. Now, enum is not a basic, but a user defined form of data type that consists of various integer values, and it functions to provide these values with meaningful names. Enum or enumerator is an integers or number with defined meaningful names.

We can define an enum in c language.

Example of Enumeration program

```
#include <stdio.h>
enum week { sunday, monday, Tuesday,
wednesday, Thursday, Friday, saturday };
int main() {
enum week today;
today = wednesday;
Printf ("Day %d", today+1);
return 0;
}
```

Output

Day 4.

Q. What is a compiler? How are they different from interpreters? Compare the features of compiler and interpreter.

Sol:- A compiler is computer software that readily translates programming language into machine code or assembly language or low-level language. It translates every program to binary that a computer readily understand and does the task that corresponds to the code.

- lexical analysis :- splitting of source code into an abstract fragment known as lexeme.
- Intermediate code generation.

Q. Write a program to print all prime numbers from 1 to 100.

Soln

```
#include <stdio.h>
void main() {
    int i,j;
for(i=1; i<=100; i++) {
    for(j=2; j<i; j++) {
    if(i%j==0)
        break;
    }
if(i==j)
    printf("%d", i);
}
}
```

Output

1, 2, 3, 4, 5, ---- 100.