Multiple Choice Questions 1 Mark

Q 1. What is the base data type of a pointer variable by which the memory would be allocated to it?

(a) int (b) float
(c) No datatype (d) unsigned int

Sol. (*d*) unsigned int

Q 2. In C a pointer variable to an integer can be created by the decalaration

(a) int p*; (b) int *p; (c) int +p; (d) int $p;

Sol. (*b*) int *p;

Q 3. A pointer variable can be

(a) Passed to a function
(b) Changed within a function
(c) Returned by a function
(d) Can be assigned an integer value

Sol. (c) Returned by a function

Q 4. What is wild pointer?

(a) Pointer which is wild in nature
(b) Pointer which has no value
(c) Pointer which is not initialised
(d) None of the above

Sol. (*c*) Pointer which is not initialised

Q 5. In order to fetch the address of the variable we write preceding ..... sign before variable name.

(a) Percent(%) (b) Comma(,)
(c) Ampersand(&) (d) Asterisk(*)

Sol. (*c*) Ampersand(&)

Q 6. Address stored in the pointer variable is of type .....

(a) Integer (b) Float
(c) Array (d) Character

Sol. (*a*) Integer

Q 7. Which of the following is the correct ways of declaring a float pointer:

(a) float ptr; (b) float *ptr;
(c) *float ptr; (d) None of the above

Sol. (*b*) float *ptr;

Q 8. What will be the output of the program.

```
void main()
{
   printf("%d %d", sizeof(int *), sizeof(int **));
}
```

(a) 2,4 (b) 2,2 (c) 4, 4 (d) 2,8

Sol. (*b*) 2,2

Very Short Answer Type Questions 1 Mark

Q 1. What is a pointer?

Sol. Pointer is a variable that stores the address of another variable. The syntax of a pointer is represented as

```
Data_Type * Pointer_Name;
```

An example to understand this syntax:

```
int *ptr;
```

Here, the ptr is an integer type of pointer.

Q 2. What is a Null Pointer?

Sol. A null reference or null pointer is a value saved for indicating that the reference or pointer does not refer to a valid object. They can be used to stop indirection in a recursive Data Structure, as an error value or as a sentinel value.

Q 3. What is the size of a generic pointer?

Sol. For a system of 16-bit, the size of a generic pointer is 2 bytes. If the system is 32-bit, the size of a generic pointer is 4 bytes. If the system is 64-bit, the size of a generic pointer is 8 bytes.

Q 4. Find the output of the following program?

```
void main()
{
   char *msg = "hi";
   printf(msg);
}
```

Sol. hi

Q 5. How many pointers can point to the same address?

Sol. Multiple pointers can point to the same memory address.

Q 6. Where is the pointer variable stored in memory?

Sol. Pointer variables are stored in stack memory.

Q 7. What is an array of pointers?

Sol. It is an array of the pointer variables. It is also known as pointer arrays. **Declaration**

```
data_type *name_of_array[array_size];
```

Example

```
int *a[5];
```

Here "a" is an array of 5 integer pointers.

Q 8. What does the below declaration mean?

```
int (*ptr)[10];
```

Sol. It means ptr is a pointer to an array of 10 integers.

Short Answer Type I Questions

2 Marks

Q 1. Describe the usage of pointers in C.

Sol. Some of the areas where pointers are used are:

- To access array elements
- Used to return multiple values
- Used in Dynamic memory Allocation
- To pass arguments by reference
- Reduces the execution time of the program
- Pointers help us to build complex data structures like a linked list, stack, queues, trees, and graphs.

Q 2. What is a dangling pointer in C?

Sol. A pointer pointing to a non-existing memory location is a dangling pointer. A dangling pointer is a pointer that has a value (not NULL) which refers to some memory that is not valid or does not exists.

Q 3. What is a generic pointer? When can we use a generic pointer?

Sol. When a variable is declared void type it is known as a generic pointer. It is a pointer that can point to any data.

They are used when we want to return such a pointer which applies to all types of pointers. They can also be used to increase the re-usability of the pointer.

Q 4. Explain the term double-pointer.

Sol. If a pointer holds another pointer's address, then such pointer is known as double-pointer or pointer-to-pointer.

Example

```
int **point;
```

Here, the point is a double-pointer.

Q 5. What is a wild pointer?

Sol. A pointer that is not initialised properly before its first use is known as the wild pointer. They are called so because the uninitialized pointer's behavior is undefined as it may point to some arbitrary location that can cause the program to crash. Generally, compilers warn about the wild pointer.

Q 6. Explain the meaning of the below declarations?

```
1. const int ptr;
2. const int *ptr;
3. int * const ptr;
4. int const * a const;
```

Sol. Here,

1. The "ptr" is a constant integer.
2. Here "ptr" is a pointer to a const integer, the value of the integer is not modifiable, but the pointer is not modifiable.
3. Here "ptr" is a constant pointer to an integer which means that the value of the pointed integer is changeable, but the pointer is not modifiable.
4. Here "a" is a const pointer to a const integer which means the value of the pointed integer and pointer both cannot be changed.

Q 7. What does it mean when a pointer is used inside an if statement?

Sol. A pointer can be used in a for, if, while, or do/while statement, or in any conditional expression. It prevents code from crashing.

Syntax

```
if ( condition is true )
{
   /*Run when valid address */
}
else
{
   /*When NULL pointer*/
}
```

Short Answer Type II Questions

4 Marks

Q 1. What is Dereference or Indirection Operator (*)?

Sol. Dereference operator is a unary operator that is used in the declaration of the pointer and accesses a value indirectly, through a pointer. The operand of such operator should be a pointer and the result of the operation is value addressed by the operand (pointer). For example,

```
int *iPointer; // Use of indirection operator in the declaration of pointer
```

a = *iPointer; //Use of indirection operator to read the value of the address pointed by the pointer
*iPointer = a; //Use of indirection operator to write the value to the address pointed by pointer.

Q 2. How do you dereference a pointer?

Sol. The operator * is used to do this, and is called the dereferencing operator. Dereferencing a pointer means getting the value, stored in the memory location pointed by the pointer.

Example

```
int a=5;
int* p=&a;
printf("%d",*p); //dereferencing the pointer p.
```

Q 3. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
   int i = 5;
   void *ptr;
   ptr = &i;
   printf("\nValue of iptr = %d ", *ptr);
   return 0;
}
```

Sol. It will give garbage value as void type cannot be dereferenced without typecasting.

Q 4. What will be output of following program?

```
#include<stdio.h>
int main()
{
   int a = 10;
   void *p = &a;
   int *ptr = p;
   printf("%u",*ptr);
   return 0;
}
```

Sol. 10

Q 5. What will be printed as the result of the operation below?

```
#include <stdio.h>
int main()
{
   char *p1;
   char *p2;
   p1=(char*)malloc(25);
   p2=(char*)malloc(25);
   strcpy(p1,"Coding");
   strcpy(p2,"output");
   strcat(p1,p2);
   printf("%s",p1);
}
```

Sol. **Output :** Codingoutput

Q 6. What will be output of following program?

```
#include<stdio.h>
int main()
{
   int a = 5,b = 10,c;
   int *p = &a,*q = &b;
   c = p-q;
   printf("%d",c);
   return 0;
}
```

Sol. 1

Q 7. What is the output of the below program? The assumed size of char, int, and double is 1,4,8.

```
#include<stdio.h>
int main()
{
   int a, b, c;
   char *p = 0;
   int *q = 0;
   double *r = 0;
   a = (int)(p + 1);
   b = (int)(q + 1);
   c = (int)(r + 1);
   printf("%d %d %d",a, b, c);
   return 0;
}
```

Sol. 1,4,8

Long Answer Type Questions

8 Marks

Q 1. Write a program in C to add two numbers using pointers.

Test Data :

Input the first number : 5

Input the second number : 6

Expected Output :

The sum of the entered numbers is : 11

Sol.

```
#include <stdio.h>
int main()
{
   int fno,sno,*ptr,*qtr, sum;
   printf("\n\n Pointer : Add two numbers :\n");
   printf("-------------------------------\n");
   printf(" Input the first number : ");
                                   scanf("%d",&fno);
   printf(" Input the second number : ");
                                   scanf("%d",&sno);

   ptr=&fno; qtr=&sno;
   sum=*ptr+*qtr;
   printf(" The sum of the entered numbers is :
                                        %d\n\n",sum);

   return0;
}
```

Q 2. Write a program in C to add numbers using call by reference.

Test Data :

Input the first number : 5

Input the second number : 6

Expected Output :

The sum of 5 and 6 is 11

Sol.

```
#include<stdio.h>
long addTwoNumbers(long*,long*);
int main()
{
   long fno,sno, sum;
   printf("\n\n Pointer : Add two numbers using call
                                   by reference:\n"
```

```
    printf("-----------------------------------\n");
    printf(" Input the first number : ");
                                        scanf("%ld",&fno);
    printf(" Input the second number : ");
       scanf("%ld",&sno); sum=addTwoNumbers(&fno,&sno);
          printf(" The sum of %ld and %ld is %ld\n\n",
                                        fno,sno, sum);
    return0;
}
long addTwoNumbers(long*n1,long*n2)
{
    long sum;
    sum=*n1 +*n2;
    return sum;
}
```

Q 3. Write a program in C to calculate the length of a string using a pointer.

Test Data :

Input a string : Arihant

Expected Output :

The length of the given string Arihant is : 7

Sol.

```
#include <stdio.h>
int calculateLength(char*);
void main()
{
    char str1[25];
    int l;
    printf("\n\n Pointer : Calculate the length of the
                                        string :\n");
    printf("-----------------------------------\n");
    printf(" Input a string : ");
    fgets(str1, sizeof(str1), stdin);
    l = calculateLength(str1);
    printf(" The length of the given string %s is :
                        %d ", str1, l-1); printf("\n\n");

}
int calculateLength(char* ch)
   // ch = base address of array str1 ( &str1[0] )
{
    int ctr = 0;
    while (*ch != '\0')
    {
       ctr++;
       ch++;
    }
    return ctr;
}
```

Q 4. Write a program in C to find the factorial of a given number using pointers.

Test Data :

Input a number : 5

Expected Output :

The Factorial of 5 is : 120

Sol.
```
#include <stdio.h>
void findFact(int,int*);
int main()
{
   int fact;
   int num1;
   printf("\n\n Pointer : Find the factorial of a
                                    given number :\n");
   printf("-----------------------------------\n");
   printf(" Input a number : ");
   scanf("%d",&num1);
   findFact(num1,&fact);
   printf(" The Factorial of %d is : %d
                                    \n\n",num1,fact);
   return 0;
}
void findFact(int n,int *f)
{
   int i;
   *f =1;
   for(i=1;i<=n;i++)
   *f=*f*i;
}
```

Q 5. Write a program in C to count the number of vowels and consonants in a string using a pointer.

Test Data :

Input a string: string

Expected Output :

Number of vowels : 1

Number of constant : 5

Sol.
```
#include <stdio.h>
int main()
{
   char str1[50];
   char *pt;
   int ctrV,ctrC;
   printf("\n\n Pointer : Count the number of vowels
                                    and consonants :\n");
   printf("-----------------------------------\n");
   printf(" Input a string: ");
   fgets(str1, sizeof(str1), stdin);
   pt=str1;
   ctrV=ctrC=0;
   while(*pt!='\0')
   {
      if(*pt=='A' ||*pt=='E' ||*pt=='I' ||*pt=='O'
         ||*pt=='U' ||*pt=='a' ||*pt=='e' ||*pt=='i'
                    ||*pt=='o' ||*pt=='u') ctrV++;

      else
         ctrC++;
         pt++;
   }
   printf(" Number of vowels : %d\n Number of
                    consonants : %d\n",ctrV,ctrC-1);

   return 0;
}
```