SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Coimbatore – 641035.

B.E / B. Tech – Internal Assessment Exam – II

A

Academic Year 2022 – 2023 (ODD)

FIRST SEMESTER (REGULATION R2019)

19CST101 – PROGRAMMING FOR PROBLEM SOLVING

ANSWER KEY

PART A

1. Define Data type in C. Give Example

Data type specifies the type of data that the variable can store like integer, character, floating, double, etc. The data type is a collection of data with values having fixed values, meaning as well as its characteristics.

Eg:

int: used to store integers, such as
'int myInt = 10;'
float: used to store floating point numbers, such as
'float myFloat = 3.14;'

2. Distinguish between while and do while statement in C.

While	Do While
It checks the condition first and then executes statement(s)	This loop will execute the statement(s) at least once, then the condition is checked.
While loop allows initialization of counter variables before starting the body of a loop.	Do while loop allows initialization of counter variables before and after starting the body of a loop.
It is an entry controlled loop.	It is an exit controlled loop.

3. Show array declaration and initialization with example.

An initializer list initializes elements of an array in the order of the list. For example, consider the below snippet:

int arr[5] = {1, 2, 3, 4, 5};

This initializes an array of size 5, with the elements {1, 2, 3, 4, 5} in order

4. Declare a two dimensional array which receives 10 rows & 25 columns.

int myArray[10][25];
This code declares a 2D array named myArray with 10 rows and 25 columns. Each element of the array is an integer.

5. Define String and use of '\0' character in string.

In C, a string is an array of characters terminated by a null character, represented by the '\0' character. The '\0' character is used to indicate the end of the string and is automatically added to the end of any string literal.

char myString[] = "hello";

PART B

6. (a). Develop a C program to implement all operators concept and explain it.

```
#include <stdio.h>
#include<conio.h>
int main()
{
// Arithmetic Operators
int a = 5, b = 3, c;
c = a + b; // addition operator
printf("%d + %d = %d\n", a, b, c);
c = a - b; // subtraction operator
printf("%d - %d = %d\n", a, b, c);
c = a * b; // multiplication operator
printf("%d * %d = %d\n", a, b, c);
c = a / b; // division operator
printf("%d / %d = %d\n", a, b, c);
c = a % b; // modulus operator
printf("%d %% %d = %d\n", a, b, c);
++a; // increment operator
printf("a incremented by 1: %d\n", a);
--b; // decrement operator
printf("b decremented by 1: %d\n", b);

// Relational Operators
int x = 5, y = 3;
if (x == y) { // equal to operator
printf("%d is equal to %d\n", x, y);
} else {
printf("%d is not equal to %d\n", x, y);
}
if (x != y) { // not equal to operator
printf("%d is not equal to %d\n", x, y);
} else {
printf("%d is equal to %d\n", x, y);
```

```
}
if (x > y) { // greater than operator
printf("%d is greater than %d\n", x, y);
} else {
printf("%d is not greater than %d\n", x, y);
}
if (x < y) { // less than operator
printf("%d is less than %d\n", x, y);
} else {
printf("%d is not less than %d\n", x, y);
}
if (x >= y) { // greater than or equal to operator
printf("%d is greater than or equal to %d\n", x, y);
} else {
printf("%d is not greater than or equal to %d\n", x, y);
}
if (x <= y) { // less than or equal to operator
printf("%d is less than or equal to %d\n", x, y);
} else {
printf("%d is not less than or equal to %d\n", x, y);
}

// Logical Operators
int p = 1, q = 0;
if (p && q) { // logical AND operator
printf("p && q is true\n");
} else {
printf("p && q is false\n");
}
if (p || q) { // logical OR operator
printf("p || q is true\n");
} else {
printf("p || q is false\n");
}
if (!p) { // logical NOT operator
printf("!p is true\n");
} else {
printf("!p is false\n");
}

// Bitwise Operators
int r = 5, s = 3, t;
t = r & s; // bitwise AND operator
printf("%d & %d = %d\n", r, s)
t = r | s; // bitwise OR operator
printf("%d | %d = %d\n", r, s)
t = r ^ s; // bitwise EX-OR operator
printf("%d ^ %d = %d\n", r, s)
// bitwise Complement operator
printf("~ %d = %d", t, ~t);
```

```
// bitwise Left – shift operator
printf(“The value of t<<2 is : %d ", t<<2);
// bitwise Right – shift operator
printf(“The value of t>>2 is : %d ", t>>2);

return 0;
}
```

(b). Build a C program to calculate the sum of first n natural numbers using while, do while & for loop.

```
#include <stdio.h>
#include<conio.h>
int main()
{
int n, i, sum = 0;
printf("Enter the value of n: ");
scanf("%d", &n);

// Using while loop
i = 1;
while (i <= n)
{
sum += i;
i++;
}
printf("Sum using while loop: %d\n", sum);

// Using do-while loop
i = 1;
sum = 0;
do
{
sum += i;
i++;
} while (i <= n);
printf("Sum using do-while loop: %d\n", sum);

// Using for loop
sum = 0;
for (i = 1; i <= n; i++)
{
sum += i;
}
```

```
printf("Sum using for loop: %d\n", sum);
return 0;
}
```

Output:

Enter the value of n: 5

Sum using while loop: 15

Sum using do while loop: 15

Sum using for loop: 15

7. (a). i. Construct a C program to perform matrix multiplication for the matrix size 3 X 3.

```
# include<stdio.h>
#include<conio.h>
void main()
{
clrscr();
int a[10][10], b[10][10],c[10][10];
int I,j,m,n,p,q,k;
printf(“give the row and column of a matrix A\n”);
scanf(“%d%d”,&n,&m);
print f (“give the row and coloumn of a matrix B\n”);
scanf (“%d%d”,&p,&q);
if(m==p)
{
printf(“matrices can be multiplied \n”);
printf(“enter the elements of the A matrix \n”);
for (i=0; i<n; i++)
for(j=0; j<m; j++)
scanf(“%d”,&a[i][j]);
printf(“enter the elements of b matrix\n”);
for(i=0; i<p; i++)
for(j=0;j<q; j++)
scanf(“%d”,& b[i][j]);
for(i=0; i<n; i++)
```

```
{
for(j=0; j<q; j++)
{
c[i][j]=0;
for(k=0; k<m;k++)
{
c[i][j]=c[i][j]+a[i][k]*b[k][j];
}
}
}
}
else
printf(“matrix can not be multiply”);
for (i=0; i<n; i++)
for(j=0;j<m;j++)
printf(“%d\t”,c[i][j]);
}
```

Output

```
Give the row and column of a matrix A
2
2
Give the row and column of a matrix B
2
2
Matrices can be multiplied
Enter the elements of the A matrix
1
2
3
4
Enter the elements of B matrix
5
6
```

7

8

19 22 43 50

ii. List the Characteristics and advantages of Array.

Characteristics:
Homogeneous: In C, an array can store elements of the same data type.
Fixed Size: Once an array is created in C, its size cannot be changed.
Contiguous Memory Allocation: Array elements in C are stored in contiguous memory locations.
Indexed: Array elements in C can be accessed using an index starting from 0.

Advantages:
Efficiency: Arrays in C are highly efficient for accessing and manipulating data due to their contiguous memory allocation and indexed access.
Easy to Implement: Arrays are relatively easy to implement in C, making them a popular choice for many programming applications.
Flexibility: Arrays in C can be used to store a variety of data types, including primitive data types, structures, and user-defined types.
Random Access: In C, arrays provide random access to the elements, allowing for efficient searching and sorting algorithms.

(b). i. Build a C Program to find the number of distinct elements in a sorted array.

```
#include <stdio.h>
int main() {
int arr[100], n, count = 1;
// Input the size of array and elements
printf("Enter the size of the array: ");
scanf("%d", &n);
printf("Enter the elements of the array in sorted order: ");
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
// Count the number of distinct elements in the array
for (int i = 1; i < n; i++) {
if (arr[i] != arr[i-1]) {
```

```
count++;
}
}
printf("The number of distinct elements in the array is %d\n", count);
return 0;
}
```

Output

Enter the size of the array: 9

Enter the elements of the array in sorted order: 1 1 2 3 3 3 4 4 5

ii. Choose the string functions to find the length of given string S1= “welcomtosnsct”.

```
#include <stdio.h>
#include <string.h>
int main() {
char s1[ ] = "WELCOME TO SNSCT";
int length = strlen(s1);
printf("%d", length);
return 0;
}
```

Output : 17

8. (a). (i) An Armstrong number is a three-digit integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3^3 + 7^3 + 1^3 = 371. Write a c program to find whether a given number 417 is an Armstrong number or not.

An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.

For example, 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153.

```
#include <stdio.h>
#include <math.h>
int main()
{
int number, originalNumber, remainder, result = 0, n = 0;
printf("Enter an integer: ");
scanf("%d", &number);
originalNumber = number;
// count the number of digits in the given number
while (originalNumber != 0)
```

```
{
originalNumber /= 10;
++n;
}
originalNumber = number;
// find the sum of each digit raised to the power of the number of digits
while (originalNumber != 0)
{
remainder = originalNumber % 10;
result += pow(remainder, n);
originalNumber /= 10;
}
if (result == number)
printf("%d is an Armstrong number.", number);
else
printf("%d is not an Armstrong number.", number);
return 0;
}
```

Output

417 is not an Armstrong number.

Since 4^3 + 1^3 + 7^3 = 192, which is not equal to the original number 417, it is not an Armstrong number.

(ii) Utilize the looping concept to generate the given Patterns

1

10

101

1010

10101

101010

1010101

10101010

101010101

1010101010

```
#include <stdio.h>
int main() {
int i, j;
for (i = 1; i <= 10; i++) {
for (j = 1; j <= i; j++) {
if (j % 2 == 0) {
printf("0");
} else {
printf("1");
}
}
printf("\n");
}
return 0;
}
```

(b) Ram and Seetha studying in SNS college of Technology, The professor Anuman want to update the details of Ram & Seetha in ERP portal so that the professor want to perform the string operations with the following strings.

String1=SNSCT

String2=DEPARTMENT OF CSE

strcat(), strcmp(), strncpy(), strrev(),strupr(),strlwr(),strlen()

```
#include <stdio.h>

#include <string.h>

int main()

{

char str1[] = "SNSCT";

char str2[] = "DEPARTMENT OF IT";

char str[100]; // For use in functions that modify strings

// strcit(): compares two strings case-insensitively

int cit = strcasecmp(str1, "SNSCT");

printf("The result of strcasecmp is %d\n", cit);
```

```
// strcmp(): compares two strings case-sensitively
int cmp = strcmp(str1, "SNSCT");
printf("The result of strcmp is %d\n", cmp);
// strcpy(): copies one string to another
strcpy(str, str1);
printf("str1: %s\nstr: %s\n", str1, str);
// strncpy(): copies a specified number of characters from one string to another
strncpy(str, str2, 5);
printf("str2: %s\nstr: %s\n", str2, str);
// strrev(): reverses a string
strcpy(str, str1);
strrev(str);
printf("str1: %s\nstr: %s\n", str1, str);
// strupr(): converts a string to uppercase
strcpy(str, str1);
strupr(str);
printf("str1: %s\nstr: %s\n", str1, str);
// strlwr(): converts a string to lowercase
strcpy(str, str2);
strlwr(str);
printf("str2: %s\nstr: %s\n", str2, str);
// strlen(): finds the length of a string
int len = strlen(str2);
printf("The length of str2 is %d\n", len);
```

return 0;

}

Output