NAME: R SAI PRANAV AADHITYA
REG NO: 2102000024
SEMESTER 3
SCHOOL OF COMPUTING AND DATA SCIENCE
EMAIL: saipranavaadhitya.r-25@scds.saiuniversity.edu.in

ITP

Assignment 02

PART - A

1. Out of the following, which one is not a valid keyword to define a data type for a variable

a) int
b) float
c) real
d) double

2. In terms of memory usage by various C variable types, find the correct order

a) char > int > float
b) char < int < double
c) double > char > int
d) int > char > float

3. Which of the following is not an operator in C?

a) ++
b) !
c) sizeof()
d) None of the above

4. Which of the following is not an identifier?

a) cat!
b) c_a_t_!
c) !cat
d) _CAT!

5. What is the equivalent c statement for the following?

x -= y+1

(a) x = x-y-1

b) x = x-y+1

c) x = -x-y-1

d) x = x+y-1

6. What is the correct order of evaluation for the expression

z = x+y* 3 /4 % 2 -1

a) * / % = + -

(b) * / % + - =

c) / * % - + =

d) - + = * % /

7. If addition had higher precedence than multiplication, what will be the value of the following expression

x = 1+2* 3 + 4* 5

a) 27

b) 47

c) 69

(d) 105

8. How many times will the following loop execute?

```
for (j = 1; j <= 10; j = j - 1
```

(a) Forever

b) Never

c) 0

d) 1

9 If no return type is mentioned in the function declaration, what is the default data type assumed?

a) void

(b) int

c) float

d) char

10. Missing elements of partially initialized integer arrays are filled with ?

a) NULL

(b) 0

c) Garbage values

d) 1

PART - B

Explain with output.

11.
```
int main (void)
{
    int value = 3;
    value = value + 2 * value ++ ;
    printf ("%d \n", value );
    return 0;
}
```

Output 10

Explanation: `value + 2 * value++;` is basically doing first calculates 3 + 3*2++ - which means, it 3+3*2, which is 9, and increments one to this value with the '++' operator, and gives the value 10.

12.
```
int main (void)
{
   int value;
   value = 5;
   while (value <= 10)
   {
     printf ("%d ", value);
     if (value > 7)
        break;
     value ++;
   }
   printf ("%d \n", value+10);
   return 0;
}
```

Loop output: 5 6 7 8 18

Explanation: In the while loop, the variable 'value' is being printed and then a space is given. Then, the if statement is ignored since value ≯ 7, and value is incremented by one. This keeps going on since the while loop holds for value <= 10. when value becomes 7 and value++ occurs. Hence, value is incremented and prints out 8 along with 5, 6 and 7, and since value > 7, the loop breaks

The loop didn't stop when value was 7, since value is still not greater than 7. Hence, the value continues to increment until 8, when it finally stops. After the loops are executed, value now holds the integer 8. At the end, value+10 is printed, which is 18. Hence, 5 6 7 8 18 is the output, followed by a newline.

13

```
int main(void)
{
  int value = 1;
  while (value++ <= 1)
  {
    while (value++ <= 2)
      printf ("%d ", value);
  }
  printf ("%d ", value);
  return 0;
}
```

Output: 3 5

Explanation: Initially, value = 1. The while (value++ <= 1) works in such a way that first it checks if value <= 1, and then only increments value. Now, value is 2. Inside that, there's another while loop which checks if value++ <= 2. Applying the same logic as value++ <= 1 as seen before, value is now incremented to 3, and this is printed. Now, that 3 increments to 4 in the same loop at value++, and goes to the outer while loop

since the inner loop is broken, and it again gets incremented - now from 4 to 5, and breaks out of even that outer loop since the condition value++ <=1 isn't satisfied. when it is finally printed out of all loops, hence, 3 5 is the output

14.
```
int main(void)
{
  int value;
  value = 10;
  do
  {
    ;
  } while (value++ < 10);
  while (value++ <= 11);
  printf ("%d", value);
  return 0;
}
```

Output: 13

Explanation

-) Initially, the value is 10. There is nothing inside 'do',
-) value goes to while (value++ < 10) and increments to 11
-) Moves on to the next line and increments twice until it goes to 13
-) On the final line, 13 is printed out as value

15. For the following code, write an equivalent code using while loop instead of for loop Explain how this equivalence is established.

```
int factorial (int n)
{
    int i, ret 1;
    for (i=2; i<=n; i++)
        ret *= i;
    return ret;
}
```

Equivalent using while loop:

```
int factorial_equivalent (int n)
{
    int i = 2;
    int ret = 1;
    while (i <= n) {
        ret *= i;
        i++;
    }
return ret;
}
```

Explanation:

-) using for (i=2, i<=n; i++), the equivalent using while is while (i <= n), and i++ inside towards the end of the loop. In the for loop, i is initialized as 2, and in the while loop, it is directly initialized at 2 since i++ will only be inside the loop. ret *i keeps going until i<=n.

16. Explain the following code with the correct output:

```
int main(void)
{
    float a[] = {4.3, 3.2, 2.6, 6.4, 1.9};
    printf("%ld bytes\n", sizeof(a));
    return 0;
}
```

Output: 20 bytes

Explanation: The size of a float data type is 4 bytes in C. sizeof(a) will return the size of the whole array in bytes, which is the sum of no. of bytes for each value in that array. "%ld" signifies a long unsigned int, which is the data type for sizeof(a).

PART-C

17. Logic:

→ In function char *destination_points(), the pointer is needed if an array is to be returned. dest_pts[2] is initialized as a static array, and user input is taken twice and added into the array with a for loop. Array dest_pts is returned.

→ In kilometres(), stops[8] is introduced, and pts1 and pts2 are mapped to the first & second elements of the resulting array of destination_pts(). A for loop iterates through 'stops' array and if any of the values are equal to pts1 and pts2, the index difference of pts2 and pts1 is calculated (with stops_index_1 and stops_index_2 respectively), and multiplied by 5 to get the no. of kilometers.

→ In cost_to_be_paid(), no. of tickets input is taken, and the cost per ticket is calculated with if-else if conditions based on the table given in the qn. If the reqd. conditions are satisfied, a discount of 10% is given. In int main(), the final fare (final fare = cost per ticket x no. of tickets) is printed.