CSE-2019

Alif Azam

April 10, 2025

Question-1

Q1. a)

a) Valid.
b) Valid.
c) Invalid: hyphen not acceptable inside identifier.
d) Valid.
e) Invalid: Keyword.
f) Invalid: Number ar first.
g) Valid.
h) Invalid: Keyword.
i) Valid.
j) Invalid: Keyword.

Q1. b) Rules for naming identifiers:
i) Identifiers can have letters (both uppercase lowercase letters), digits, dollar sign and underscore only.
ii) The first letter of a identifier should be a letter or an underscore or a dollar sign.
iii) White space is not permitted.

marks: Valid
return: Invalid: Keyword.
for: Invalid: Keyword.
roll#: Invalid.
$money: Valid.
Break: Valid.
_file1: Valid.

Q1. c)

```c
#include <stdio.h>

int main() {
    char grade;
    scanf("%c", &grade);
    if(grade == 'A'){
        printf("Excellent!\n");
    } else if(grade == 'B'){
        printf("Good\n");
    } else if (grade == 'C'){
        printf("Well done\n");
    } else if (grade == 'F'){
        printf("Better try again\n");
    } else {
        printf("Invalid grade\n");
    }
    return 0;
}
```

Question-2

Q2. a) Four different data types are:
i) character(declaration: char ch;)

- One byte
- scanf("%c", ch);
- printf("%c", ch);

ii) Integer(declaration: int roll;)

- Two bytes
- scanf("%d", roll);
- printf("%d", roll);

iii) Float(declaration: float fl;)

- Four bytes
- scanf("%f", fl);
- printf("%f", fl);

iv) double(declaration: double num;)

- Eight bytes
- scanf("%lf", num);
- printf("%lf", num);

Q2. b)

```c
#include<stdio.h>
int main(){
    int a, b, c, min;
    printf("Enter three values: ");
    scanf("%d%d%d", &a, &b, &c);
    min = a;
    if (b < a && b < c){
        min = b;
    } else if (c < a && c < b){
        min = c;
    }
    if (min%2 == 0){
        printf("%d is the min value, which is even.\n", min);
    } else {
        printf("%d is the minimum value, which is odd.\n", min);
    }
    return 0;
}
```

Q2. c) **Compile-time error:** Error found by the compiler before running the code is known as compile time error. Example:

```c
int main() {
    int x = "hello"; // type mismatch
    return 0;
}
```

Run-time error: Error occurs during program execution is known as run time error. Example:

```c
int main() {
    int a = 5, b = 0;
    int c = a / b; // division by zero
    return 0;
}
```

Question-3

Q3. a)

```c
#include<stdio.h>

int main(){
    int i, j, n, temp;
    scanf("%d", &n);
    temp = n;
    for(i=1; i<=n; i++){
        for(j=1; j<=temp; j++){
            printf("%d ", j);
        }
        printf("\n");
        temp--;
    }
    return 0;
}
```

Q3. b) a++(post-increment) executes after the statement. It returns the old value.
++a(pre-increment) executes before the staement. It returns the new value.

Q3. c)

```
3
5
```

That means, a=3 and b=5.

Question-4

Q4. a)

```c
#include<stdio.h>
int main(){
    int i = 33;
    int *j = &i;
    printf("Value of i: %d\n", *j);
    return 0;
}
```

Q4. b) i)double f(double a, int b);
The function f returns a double type value. It takes two values(one double and one integer) as parameters. ii)void f(long a, short b, unsigned c); The function f returns no value. It takes three values(one long integer, one short integer and one unsigned integer) as parameters. iii)char f(void); The function f returns a character type value. But takes no values as parameter.

Q4. c)

```c
#include<stdio.h>

int main(){
    int mat1[2][2]={{1, 2}, {3, 4}}, mat2[2][2]={{5, 6}, {7, 8}}, sum[2][2];
    int i, j;
    for(i=0; i<2; i++){
        for(j=0; j<2; j++){
            sum[i][j]=mat1[i][j]+mat2[i][j];
        }
    }
    for(i=0; i<2; i++){
        for(j=0; j<2; j++){
            printf("%d ", sum[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Question-5

Q5. a)

```c
#include<stdio.h>
int main(){
   int i, n;
   scanf("%d", &n);
   for(i=0; i<n;){
       if (n%2 != 0){
           i++;
       } else if (n%2 == 0){
           i+=2;
       }
       printf("%d", i);
   }
   return 0;
}
```

Q5. b) Here, roll 2 is holding place at index 1.

```c
#include <stdio.h>

struct Student {
    int roll;
    char name[50];
    int age;
};

int main() {
    struct Student students[5];
    int i;

    for (i = 0; i < 5; i++) {
        students[i].roll = i + 1;
        printf("Enter name and age for roll %d: ", students[i].roll);
        scanf("%s %d", students[i].name, &students[i].age);
    }
    printf("\nDetails of student with roll 2:\n");
    printf("Roll: %d\nName: %s\nAge: %d\n", students[1].roll, students[1].name, students[1].age);
    return 0;
}
```

Q5. c) Here, the array contains 5 fixed values.

```c
#include <stdio.h>

float average(int arr[], int size) {
    int sum = 0;
    for(int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return (float)sum / size;
}

int main() {
    int nums[] = {10, 20, 30, 40, 50};
    int n = 5;

    float avg = average(nums, n);
    printf("Average = %.2f\n", avg);

    return 0;
}
```

Question-6

Q6. a) Function prototype: A function prototype is a declaration of a function that specifies its name, return type, and the number and types of its parameters. It essentially tells the compiler about the function's interface without providing the actual code.
Prototypes inform the compiler about the function's existence and how it should be called. This allows the compiler to perform type checking and ensure that the function is used correctly before it's actually defined.
Function prototypes are typically placed at the beginning of a C file, usually before the main function. This ensures that the compiler knows about the functions before they are called within main.

```c
#include <stdio.h>

int square(int num);
int main() {
    int x = 5;
    int result = square(x);
    printf("The square of %d is %d\n", x, result);
    return 0;
}

int square(int num) {
    return num*num;
}
```

Q6. b) The code initializes a global variable 'a' to 3. A loop calls a function 'functl' five times, adding the loop counter to 'a' each time and printing the updated 'a' value. The output is: 4 6 9 13 18.

Q6. c) Structures are used when we need to group together variables of different data types under a single name. This is particularly useful when we are dealing with entities that have multiple attributes or properties.
Here is a simple structure example named course, which possess four values of different data type. But all are under the same name course.

```c
struct course {
    char dept[20];
    char code[10];
    char title[30];
    float credit;
};
```

Question-7

Q7. a)

```c
char result;
float gpa;

switch(result){
    case '*':
        gpa=4.00;
        break;
    case 'G':
        gpa=3.25;
        break;
    case 'C':
        gpa=2.00;
        break;
    default:
        gpa=0.00;
}
printf("%f", gpa);
```

Since, we are comparing with some fixed values like *, G and C, switch statement is better. Because it is easier to read in discrete cases.

Q7. b)

```c
#include<stdio.h>

float AVG(int a, int b);

int main(){
    int a, b;
    float average;
    scanf("%d%d", &a, &b);
    average = AVG(a, b);
    printf("%.2f is the average.\n", average);
    return 0;
}

float AVG(int a, int b){
    return (a+b)/2.0;
}
```

Q7. c) Return statement in a function:

i) Sends a value back to the caller function, serving the purpose of creating the function.

ii) Ends function execution immediately.

Question-8

Q8. a)

```c
#include<stdio.h>

int factorial(int n);
int main(){
    int n;
    scanf("%d", &n);
    int fact = factorial(n);
    printf("Factorial of %d is %d\n", n, fact);
    return 0;
}

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

Q8. b) Benefits of pointers over the use of arrays:

i) It helps in dynamic memory allocation, when size of the array is not known previously.

ii) In the function, passing by reference gives us power to directly modify(insert or delete) values of an array.

Programmers have to initialize any pointer-type variable prior to use it because uninitialized pointers may hold garbage values. So, setting into NULL or a valid address is a good practice.

Q8. c)

```c
struct course {
    char dept[20];
    char code[10];
    char title[30];
    float credit;
};
```