Reg. No. .....

Indian Institute of Information Technology, Surat
Department of Computer Science & Engineering
Makeup End Semester Solutions 1st Sem, 2025-26
Fundamentals of Computer and C Programming (CS101)

Timing: 10:00 AM - 1:00 PM **Date: 16 Feb 2026** **Max Marks: 100**

Instructions: Attempt all parts of a question at one place. If you do not, later parts may not be graded.

1. State whether the following statements are **true** or **false**. If the statement is false then **write the intended correct statement.** **(5x2=10)**

a. The last element of an array `Arr` of size `10` can be accessed by `Arr[-1]`. F, Arr[9]

b. The largest integer that can be stored in a `4`-byte int variable is 2^{31}. F, 2^{31} - 1(signed)

c. The `<stdio.h>` header file is required to use `malloc()`. F, <stdlib.h>

d. The function `strconcat()` is used to concat two strings. F, strcat()

e. In a nested loop, a `break;` statement inside the inner loop causes the program to immediately exit from the outermost loop. F, exit from the innermost loop

2. Choose the single correct answer for each of the following questions. **(5x2=10)**

a. What is the correct way to declare a pointer to an integer?
i. `int ptr;` ii. `int *ptr;` iii. `ptr int;` iv. `ptr int*;`

b. What happens if you try to assign a value to an array name (e.g. `arr=10;`)?
i. Changes the first element ii. Changes the array size
iii. Runtime Error iv. Compilation Error

c. What happens if you forget the break statement inside a switch case?
i. Compilation Error ii. Runtime Error
iii. Program goes into an infinite loop iv. Next case is executed (after the matching case)

d. Which of the following is "False" in a conditional statement?
i. `-1` ii. `0` iii. Both, i and ii iv. None of i or ii

e. Which of the following is the correct syntax for an Else-If ladder?
i. `elif (condition)` ii. `elseif (condition)`
iii. `else if (condition)` iv. `else (condition)`

3. For each of the C code snippets below, classify it as compilation error, runtime error, undefined behavior, or correct. Then, **fix and rewrite** the corrected code snippet (if needed) assuming standard context. **(3x5=15)**

a.
```
str wish = "good luck!" //String wish should store "good luck" (without quotation marks) Compilation error, char * wish ="good luck !" or char ch[]="good luck!" or char ch[11] = "good luck"
```

b.
```
int *ptr; *ptr = 101; //The value of integer pointed by ptr should be 101 run time error / undefined behaviour (because the integer is not yet allocated).
```

```
int *ptr; ptr = (int*) malloc(sizeof(int)); *ptr=101; or
int x=101; int *ptr=&x;
```

c. `long double x=23; scanf("%ld", x); //Read a long double from user`
`Undefined, long double x; scanf("%Lf", &x);`

4. Write the output of the following programs. Assume that the required header files are included. Assume a standard 64-bit Linux environment identical to that in labs. **(3x5=15)**

a.

```
int calc(int x, int y) {
    if (x <= 0) return y;
    if (y <= 0) return x;
    return calc(x - 2, y - 1) + x;
}

int main() {
    printf("%d", calc(5, 3));
    return 0;
}
```

Solution : 9

b.

```
void mystery(int n) {
    if (n <= 0) return;
    if (n % 2 == 0) {
        mystery(n - 1);
        printf("%d ", n);
    } else {
        printf("%d ", n);
        mystery(n - 2);
    }
}
int main() {
    mystery(4);
    return 0;
}
```

Solution 3 1 4

c.

```
int main() {
    int i = 0;
    while (i < 5) {
        i += 2;
        if (i == 4) {
            continue;
        }
        printf("%d ", i);
    }
    return 0;
}
```

Solution: 2 6

5. Answer the following: **(4x5=20)**

a. Give the function prototype of `malloc()`. Explain **in three lines** its purpose and why the function `free()` should be called after `malloc()`.
void *malloc(size_t); (instead of size_t, long long int is also permissible)

b. Show calculation and write the number `-42` as a `4`-byte integer under `2`'s complement notation. **Box your final answer.** **11111111 11111111 11111111 11010110 (FFFFFFD6)**

c. Explain the concept of a recursive function **in two lines** and write a recursive function `int sum_of_digits(int n);` that recursively computes the sum of digits of the parameter `n`.

```
int sum_of_digits(int n) //Assuming n is positive
 {
    if (n == 0) return 0;
    return sum_of_digits(n/10) + n % 10;
 }
```

d. Explain the von Neumann architecture **in two lines** and draw a basic diagram.
Instructions and data share the same memory space.

6. Write complete C programs for **any two** of the following. You should properly indent and add comments in your code. Use appropriate datatypes as necessary. **(2x15=30)**

a. Read a line of text (string) from the user, followed by a single character to be searched for. You may assume the string contains only upper and lower case alphabets and will not exceed **`1000`** characters. Implement a function with the prototype: **`void swap_case(char *str, char c);`** that swaps the case of every occurrence of **`c`** in **`str`**. The main function should then display the updated string.

```c
#include <stdio.h>
#include <string.h>
void swap_case(char *str, char c) {
  while(*str != '\0') {
    if(*str == c) *str = c^32; //Swap cases
    str++;
  }
}
int main() {
    char str[1002];
    char c;
    if (fgets(str, sizeof(str), stdin) == NULL) return 0;
    scanf(" %c", &c); //Ignore white spaces when scanning for c
    swap_case(str, c);
    printf("%s", str);
    return 0;
}
```

b. Read a number **`N`** and then an array **`Arr`** of **`N`** integers from the user. You may assume that **`N`** and all input integers are positive and at most **`1000`**. Calculate and print the mean and variance of **`Arr`**.

```c
#include <stdio.h>
int main() {
    int n, i;
    int arr[1000];
    double sum = 0.0, mean, variance = 0.0;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
        sum += arr[i];
    }
    mean = sum / n;
    for (i = 0; i < n; i++) {
        variance += (arr[i] - mean) * (arr[i] - mean);
    }
    variance /= n;
    printf("Mean: %.2f\n", mean);
    printf("Variance: %.2f\n", variance);
    return 0;

}
```

c. Implement a **`struct Time`** with **`int`** fields hours and minutes. Write a function **`add_time`** that takes two such **`Time`** structs and adds them by returning a **`struct Time`** variable. In the main function, you should call **`add_time`** function to add **`1`** hour **`30`** minutes and **`3`** hours **`40`** minutes and print the resulting time.

```c
#include <stdio.h>
typedef struct {
    int hours;
    int minutes;
} Time;
Time add_time(Time t1, Time t2) {
    Time result;
    result.minutes = t1.minutes + t2.minutes;
    result.hours = t1.hours + t2.hours + (result.minutes / 60);
    result.minutes %= 60;
    // result.hours %= 24; This is also allowed.
    return result;
}
int main() {
    Time time1 = {1, 30};
    Time time2 = {3, 40};
    Time sum = add_time(time1, time2);
     printf("The sum of the times is: %d hours and %d minutes\n",
sum.hours, sum.minutes);
    return 0;
}
```