Name: ____________________ Roll No: _________________InvSign:__________

CS1109/CS1110 (Introduction to Computing)

End SemExam 17th Nov 2025 2PM Total Marks:60

Write Precise Answer in the Space Provided in the Question Paper. Use Anwersheet for Rough Work

==

1. **[1 Mark] Declare an appropriately sized character array and copy your name into the declared variable using the `strcpy` function in C.**

   char name[20]; //size need to be higher than number of character in your name
   strcpy(name, "asahu"); //your name

2. **[2 Marks] Write IEEE-754 single-precision (32-bit) floating-point representation of 128.5**

Sign:	Exponent:	Mantissa:
0	*10000110*	*00000001 000000000000000*

   *$128.5=(10000000.1)_2=(1.00000001x2^7)_2$ ==> **Mantissa**=00000001, Exp=7+127=134*

3. **[2 Marks] Write a recursive function to find the sum of the digits of an unsigned integer.**

```
unsigned int sumOfDigits(unsigned int n) {
    if (n == 0)     return 0;     //Base Case
 return (n % 10) + sumOfDigits(n / 10);  // Recursive case: last digit + sum of remaining digits
}
```

4. **[1 Marks] Why we do not need to use "&" when taking input for a string using the `scanf` function?**

```
char  Name[25];
scanf("%s", Name);
```

   The array name `Name` automatically becomes a pointer (const ptr) to its first element. `Name` is effectively the same as: &Name[0].

5. **[2 marks] What will be the output of the following code snippet?**

```
struct S{int a,b;}; union U{int a,b;};
printf("%d %d %d", sizeof(struct S*), sizeof(union U*), sizeof(void *));
```

   8 8 8

6. **[2 Marks] What will be the output of the following code snippet?**

```
int A[10][10], i, j, *P;
for(i=0; i<10; i++)
  for(j=0;j <10; j++)  A[i][j]=i*10+j;
P=A+30;
printf("%d",  *(P+21));
```

   51

7. **[2 Marks] What will be the output of the following code snippet?**

```
int A[100]={1,4,4, 20, 1, 2, 4, 6, 10 }, *p, *q;
q=A+20; p=&A[150];
printf("%ld", q-p);
```

   -130

8. **[1+2 Marks] Analyze time complexity (and reasons) of following code for the invocation C=Fib(N).**

```
int F[N];//initialized to 0
int Fib(int n){
  if(n<=1) return n; if( F[n]!=0 ) return F[n];
       return Fib(n-1)+Fib(n-2);  //return F[n]=Fib(n-1)+Fib(n-2);
}
int main(){
          for(i=0;i<N;i++) F[i]=0;
C=Fib(N);//Invocation
}
```

As: As this code donot update the Fib Array, so it donot take benefit of Memorization, Hence the Fib Calcluation complexity is exponential 2^N (as dicussed in the class).
Instead of "return Fib(n-1)+Fib(n-2);" if "return F[n]=Fib(n-1)+Fib(n-2);" is replaced then it reduce the complexity to O(n); Effectively number of recursive it will make is very less, as it memorized. Time complexity is O(n). When it call Fib(n), it recursively call Fib(n-1) and it will set F[n-2] to non-zero and it will not make Fib(n-2) call.

9. **[2 Marks] What is the meaning of signature of a function in a C program?**

Signature of a function is Function name, return type, number and types of its parameters in order.

10. **[2 Marks] Declare a function pointer variable `fp` for the following function.**
`int myfunadd( int *p, int A) {*p+=A; return *p;}`

int (*fp)(int *, int);

11. **[1 Marks] For the `fopen` function in C, what will happen if the file `test.txt` does not exist in the same directory?**

```
FILE *fptr;
fptr= fopen ("test.txt","r");
```

fopen fails and returns NULL, It does not create the file.

It does not read or write anything.

12. **[2 Marks] Write a one-line code statement to read 100 floating-point numbers using `fread` from a file (opened in binary mode) with file pointer `fp` into the floating-point array `Float F[400]`.**

fread(F, sizeof(float), 100, fp);

13. **[1+1 Marks] What will happen if the following code is executed (or is there any chance of getting an error)?**

```
fseek(fp, -25, SEEK_CUR);
```

Move the file position 25 bytes backward from the current position.

If the current position is < 25 bytes, Then the new position would be before the beginning of the file, which is illegal. Fseek fails show runtime error. Yes, there is a chance of error.

14. **[3 Marks] What are the time complexities of the (a) search, (b) find-max, and (c) find-min operations in a height balanced binary search tree (B-BST)?**

(a) You follow one path from root to a leaf, comparing keys at each node. Time Complexity: O(log n)
(b) In a BST, the maximum element is found by repeatedly following right-child pointers until the last node. At most you follow the height of the tree.Time Complexity: O(log n)
(c) Same reasoning as find-max, but follow left-child pointers. Time Complexity: O(log n)

15. [3 Marks] **Complete the function that copies contents of a source file to destination file in text mode.**

```
void file_copy(char *src, char *dst){
int ch;  FILE *fp1 = fopen(src, "r"); FILE *fp2 = fopen(dst, "w");
if (fp1 == NULL || fp2 == NULL) {printf("Error in src/dst");}
while ((ch = fgetc(fp1)) != EOF)  fputc(ch, fp2);              //fprintf(fp2,"%c",ch);
fclose(fp1);    fclose(fp2);
}
```

16. [3 Marks] **Write a program that takes two input filenames from the command line and copies the contents of the first file to the second file in text mode. You use function defined in Q15**

```
int main(int argc, char *argv[]){

    file_copy(argv[1], argv[2]);

}
```

17. [3 Marks] **Write a small piece of code to find the number of lines in the text file `list.txt` using `fgetc()`.**

```
int ch, line_cnt=0;     FILE *fp = fopen("list.txt", "r");
while ((ch = fgetc(fp1)) != EOF)
                    if (ch == '\n') line_cnt++;
printf("Number of lines: %d\n", line_cnt);
```

18. [2 Marks] **What is a hotspot (or bottleneck) in a C code, and how can you identify it?**

Hotspot is a function, loop, or code block that consumes a large fraction of CPU time, executed very frequently, or performs expensive operations.
Use any one profiler tool to find the hotspot. These can be gprof, valgrind, perf, or any

19. [2 Marks] **What tool can be used to identify runtime errors in C code, and which compiler option should be used during compilation to enable runtime-error detection?**

Any debugger: gdb or ddd or any other debugger
GCC Compiler option: -g

20. [2 Marks] **What are the major demerits of defining functions using the `#define` preprocessor directive?**

1. No type checking: So the compiler cannot verify argument types. [MOST IMP ONE]
2. Multiple evaluation of arguments: Macro parameters may be evaluated more than once, leading to incorrect results or side effects, 3. No debug support:

21. [2 Marks] **What are the typical contents of a header file in C programming?**

Function Declarations, Macro definitions, Type and structure defintion, extern variable declarations and Include guards: #ifndef MYHEADER_H #define MYHEADER_H // declarations here #endif

22. [2 Marks] **Analyze the average (or amortized) time complexity of the insert-at-the-end operation in a size-doubling dynamic array.**

Suppose Has current capacity `C`, When full, allocates a new array of size `2C`, copies over elements, and frees old, However, the expensive resizes become rarer as the array grows. The total cost of n insertions is O(n), so:Amortized (average per operation) time complexity: O(1) for insert-at-the-end.

Total Number of resizing cost for N insert : (1+2+4+8+...+N)/N=2N/N=2 =O(1)

23. **[2 Marks] What are the major demerits of a dynamic array in terms of operation complexity?**

(a) Costly insertion/deletion at arbirary position due to all later elements must shift one step right/left.
(b) Memory overhead due to unused capacity, (c) Costly resizing (O(n))

24. **[3 Marks] Identify all the levels of freeing memory in a dynamic array and write a comment for each "free" call in the following code.**

```
void DynArray_Destroy(Stack *a, void (*free_fn)(void *)) {
for (int  i = 0; i < a->size; ++i)
      if (a->data[i]) free_fn(a->data[i]);  //Level 3: //Freeing objects the array points to
    free(a->data);                          //Level 2: //Freeing pointer array
    free(a);                                //Level 1  //Freeing DynArray Object itself
}
```

```
      free_fn(a->data[i]);           //Level 3: //Freeing objects the array points to

       free(a->data);                //Level 2: //Freeing pointer array

    free(a);                         //Level 1  //Freeing DynArray Object itself
```

25. **[4 Marks] Given an array-of-lists data structure where the size of the array is M, the number of elements to be inserted is N, and the numbers to be inserted are 10-digit numbers that are randomly distributed:**
a) What will be the expected time complexity of each insertion if the insertion is performed at the beginning of the list?

Constant time : O(1)

b) What will be the expected time complexity of the search operation?

Avg= (N/M)/2 = O(N/M)

26. **[2 Marks] What is the difference between the capacity and the size of a generic stack implemented using a dynamic array container?**

```
typedef struct Stack{
      void **data;
      int capacity,  size;
}Stack;
```

Capacity:The total number of elements the dynamic array can hold before it needs to be resized (expanded).
Size: The number of elements currently stored in the stack.

27. **[3 Marks] Implement a queue data structure using two stacks. Assume the stacks support `push` and `pop` operations and have infinite capacity. Your designed queue should support the `enqueue/push_back()` and `dequeue/pop_front()` operations.**

A queue data structure can be implemented using two stacks ***inputStack*** *and* ***outputStack****.*

- *Enqueue(Item X): To enqueue an Item X, simply push onto the* ***inputStack****. Complexity O(1).*
- *Item Dequeue():*
  - *If* ***outputStack*** *is not empty, pop the top element from* ***outputStack****. This element represents the front of the queue.*
  - *If* ***outputStack*** *is empty, transfer all elements from* ***inputStack*** *to* ***outputStack****. repeatedly popping from* ***inputStack*** *and pushing onto* ***outputStack*** *until* ***inputStack*** *is empty. If return Null if both the stacks are empty.*