Code Snippets on Files

Q1

```
FILE *fp = fopen("data.txt", "w");
fprintf(fp, "Hello World");
```

What happens when this code runs?
A) Appends data
B) Overwrites file
C) Reads file
D) Opens in binary

Q2

```
FILE *fp = fopen("log.bin", "wb");
int x = 50;
fwrite(&x, sizeof(x), 1, fp);
```

The file created is:
A) Text file
B) Binary file
C) CSV file
D) PDF file

Q3

```
FILE *fp = fopen("info.txt", "r");
if(fp == NULL) printf("Error");
```

Error prints when:
A) File is empty
B) File doesn't exist

Dr.Kavitha N
Dept. of IT

C) File has spaces
D) Always

Q4

```
FILE *fp = fopen("data.txt", "r");
char ch = fgetc(fp);
ungetc(ch, fp);
ch = fgetc(fp);
printf("%c", ch);
```

The printed character is:
A) First character
B) Second character
C) Third character
D) Undefined

Q5

```
FILE *fp = fopen("a.txt", "a");
fputs("A", fp);
```

Opening with a means:
A) Overwrite file
B) Append data
C) Read only
D) Delete file

Q6

```
FILE *fp = fopen("file.txt", "r");
char s[10];
fscanf(fp, "%s", s);
printf("%s", s);
```


Dr.Kavitha N
Dept. of IT

%s in fscanf stops at:
A) Tab
B) Newline
C) Space
D) End of file

Q7

```
FILE *fp = fopen("text.txt", "r");
fseek(fp, 0, SEEK_END);
long size = ftell(fp);
printf("%ld", size);
```

This prints:
A) Characters count
B) Words count
C) File size in bytes
D) Lines count

Q8

```
FILE *fp = fopen("abc.txt", "r+ ");
```

r+ mode means:
A) Read only
B) Read + Write (no overwrite)
C) Write only
D) Append only

Q9

```
FILE *fp = fopen("file.txt", "r");
char ch;
while((ch = fgetc(fp)) != EOF)
```


Dr.Kavitha N
Dept. of IT

```
printf("%c", ch);
```

This loop:
A) Reads entire file
B) Reads first line only
C) Reads last 5 chars
D) Prints only numbers

Q10

```
FILE *fp = fopen("x.txt", "r");
fseek(fp, -1, SEEK_END);
```

-1 from SEEK_END moves:
A) Forward one byte
B) Backward one byte
C) To middle
D) To start

Q11

```
FILE *fp = fopen("hey.txt", "w+");
```

w+ mode:
A) Read only
B) Read/write + truncate file
C) Append only
D) Binary write

Q12

```
char s[20];
fgets(s, 20, fp);
```


Dr.Kavitha N
Dept. of IT

fgets stops when:
A) Newline or (size-1)
B) Only on EOF
C) Always 20 chars
D) After a tab

Q13

FILE *fp = fopen("x.bin", "rb");

rb means:
A) Read text
B) Read binary
C) Read + Write text
D) Append binary

Q14

FILE *fp = fopen("file.txt", "r");
fgetc(fp);
long p = ftell(fp);

After reading one character, p is usually:
A) 0
B) 1
C) 2
D) -1

Q15

fwrite(arr, sizeof(int), 5, fp);

Total bytes written:
A) 5

Dr.Kavitha N
Dept. of IT

B) 10
C) 20
D) 40

Q16

FILE *fp = fopen("doc.txt", "r");
feof(fp);

feof becomes true:
A) When file opens
B) After reading past last char
C) Before any read
D) Only if file is empty

Q17

FILE *fp = fopen("temp.txt", "r");
rewind(fp);

rewind moves pointer to:
A) End
B) Middle
C) Start
D) Next line

Q18

FILE *fp;
fp = fopen("a.txt", "r");

fp is a:
A) int*
B) FILE*

Dr.Kavitha N
Dept. of IT

C) double*
D) void*

Q19

remove("old.txt");

This function:
A) Deletes file
B) Renames file
C) Clears content
D) Creates file

Q20

Appending always writes:
A) At beginning
B) At end
C) Random
D) Middle of file

Code Snippets on Searching

Q1. What is the output?

```
int a[] = {3, 6, 9, 12};

int key = 9, i;

for(i = 0; i < 4; i++)

   if(a[i] == key) break;

printf("%d", i);
```


Dr.Kavitha N
Dept. of IT

A) 1
B) 2
C) 3
D) 4

Q2. Linear search stops when?

```
for(int i=0;i<n;i++)
   if(a[i] == key)
      return i;
```

A) After full scan only
B) When middle element is found
C) When matching element is found
D) Before scanning

Q3. Output?

```
int a[]={1,4,7,10};
int key=5, i;
for(i=0;i<4;i++)
   if(a[i]==key) break;
printf("%d", i);
```

A) 4
B) 3
C) 2
D) 0

Q4. Binary search requires?

```
int a[]={5,2,7,1,9};
```

A) Any order
B) Descending only

Dr.Kavitha N
Dept. of IT

C) Sorted array
D) Unique values only

Q5. Output?

```
int a[]={2,4,6,8,10};

int low=0, high=4, mid;

mid = (low+high)/2;

printf("%d", a[mid]);
```

A) 4
B) 6
C) 8
D) 10

Q6. Binary search loop condition?

```
while(low <= high)
```

A) Incorrect
B) Correct
C) Only < allowed
D) Only >= allowed

Q7. Output?

```
int a[]={1,3,5,7};

int key=7, low=0, high=3;

int mid=(low+high)/2;

printf("%d", mid);
```

A) 1
B) 2
C) 3
D) 0

Dr.Kavitha N
Dept. of IT

Q8. When linear search fails?

```
if(i == n)
   printf("Not found");
```

A) Found at last index
B) Found at first index
C) Key never matched
D) Error always

Q9. Output?

```
int a[]={2,4,6,8};
int key=8,i;
for(i=0;i<4;i++) if(a[i]==key) break;
printf("%d", i);
```

A) 1
B) 2
C) 3
D) 4

Q10. Binary search mid formula?

A) (low+high)/2
B) (low*high)/2
C) (low-high)/2
D) low+1

Q11. Output of unsuccessful binary search?

```
if(low > high)
   printf("Not found");
```

A) Always prints
B) Prints only on failure

C) Never prints
D) Print error

Q12. Linear search time complexity?

A) O(1)
B) O(log n)
C) O(n)
D) $O(n^2)$

Q13. Binary search time complexity?

A) O(1)
B) O(log n)
C) O(n)
D) O(n log n)

Q14. Worst case linear search occurs when?

A) First element matches
B) Middle element matches
C) Last element matches
D) No element matches

Q15. What is printed?

```
int a[]={11,22,33,44};

int key=33,i;

for(i=0;i<4;i++) if(a[i]==key) break;

printf("%d", a[i]);
```

A) 11
B) 22
C) 33
D) 44

Dr.Kavitha N
Dept. of IT

Code Snippets on Sorting

Q1. What is printed?

```
int a[]={5,3,1};

for(int i=0;i<2;i++)

   for(int j=0;j<2-i;j++)

      if(a[j] > a[j+1]){

         int t=a[j]; a[j]=a[j+1]; a[j+1]=t;

      }

printf("%d", a[0]);
```

A) 5
B) 3
C) 1
D) 0

Q2. Insertion sort shifts elements until?

```
while(j>=0 && a[j] > key)
```

A) Condition fails
B) j becomes 0
C) Key becomes smaller
D) Always true

Q3. Output?

```
int a[]={4,1,3};

int min=0;

for(int j=1;j<3;j++)

   if(a[j] < a[min]) min=j;
```


Dr.Kavitha N
Dept. of IT

printf("%d", min);

A) 0
B) 1
C) 2
D) 3

Q4. Bubble sort performs?

A) One pass only
B) Multiple passes until sorted
C) Binary comparison
D) Recursion only

Q5. Output?

int a[]={9,2,6,1};

int min=0,t;

for(int j=1;j<4;j++)

if(a[j] < a[min]) min=j;

printf("%d", a[min]);

A) 9
B) 6
C) 2
D) 1

Q6. In insertion sort, key is?

key = a[i];

A) Largest element
B) Element to insert in sorted left part
C) Temporary max
D) Swapped value

Dr.Kavitha N
Dept. of IT

Q7. Selection sort swaps?

A) Every time condition matches
B) Only once per pass
C) Never swaps
D) Randomly swaps

Q8. Output?

```
int a[]={3,2,1};

int t=a[0]; a[0]=a[2]; a[2]=t;

printf("%d", a[0]);
```

A) 1
B) 2
C) 3
D) 0

Q9. Bubble sort algorithm compares?

A) Every element with last
B) Adjacent elements
C) Middle and ends
D) Only first two

Q10. Output?

```
int a[]={8,5,7};

int key=a[2];

printf("%d", key);
```

A) 5
B) 7
C) 8
D) 0

Dr.Kavitha N
Dept. of IT

Q11. Selection sort finds?

A) Next maximum only
B) Next minimum only
C) Random element
D) Element nearest to mid

Q12. Complexity of bubble sort?

A) O(log n)
B) O(n)
C) O(n log n)
D) $O(n^2)$

Q13. In insertion sort worst-case occurs when?

A) Already sorted
B) Reverse sorted
C) Random order
D) All equal

Q14. Output?

```
int a[]={10,20,30};

int t=a[1]; a[1]=a[2]; a[2]=t;

printf("%d", a[1]);
```

A) 10
B) 20
C) 30
D) 40

Q15. Selection sort first pass selects?

A) Middle element
B) Largest element

Dr.Kavitha N
Dept. of IT

C) Smallest element
D) Second largest

Dr.Kavitha N
Dept. of IT