CAT-II ACP KEY

1. a Re-write the given code snippet below using array of pointers only **(5 marks)**

```
#include <stdio.h>

const int UPPER = 3;

int main () {

   int  a[] = {10, 100, 200};
   int i;

   for (i = 0; i < UPPER; i++) {
      printf("Value of a[%d] = %d\n", i, a[i] );
   }

   return 0;
}
```

```
#include <stdio.h>

const int UPPRER = 3;

int main () {

   int  a[] = {10, 100, 200};
   int i, *ptr[MAX];

   for ( i = 0; i < UPPER; i++) {
      ptr[i] = &a[i]; /* assign the address of integer. */
   }

   for ( i = 0; i < MAX; i++) {
      printf("Value of a[%d] = %d\n", i, *ptr[i] );
   }

   return 0;
}
```

b. Fill in the blanks using array of function pointer declaration and function call statement. **(5 marks)**

```
#include <stdio.h>
void function_ add(int a, int b)
{
   printf("Addition of two number  is is %d\n", a+b);
}
void function_subtract(int a, int b)
{
   printf("Subtraction of two number is %d\n", a-b);
}
void function_multiply(int a, int b)
{
   printf("Multiplication of two number is %d\n", a*b);
}


int main()

{

   // Declare array of function pointers and initialize it here, to hold the address of this
three function given above

  void (*fun_ptr_arr[])(int, int) = {add, subtract, multiply};

________________________________________________________

   int ch, a = 15, b = 10;

   printf("Enter Choice: 0 for addition, 1 for subtraction  and 2 for multiplication\n");

   scanf("%d", &ch);

    if (ch > 2) return 0;

(*fun_ptr_arr[ch])(a, b);

      ____________________// Write the Function call statement here using array of
function pointer.

    return 0;

}
```

c. Match the following (types of pointers)

Column-A	Column-B

int *ptr = (int *)malloc(sizeof(int));
  ptr = 12; / Assume malloc doesn't return NULL) */
	Where ptr becomes a Wild pointer

int *function_1()
{
    int y = 5;
 	
    return &y;
}
 	
    int *ptr = function_1();
	Where ptr becomes a Dangling pointer

int i = 4;
    float f = 5.5;
 	
    void *ptr;
    ptr = &i;
ptr = &f
	Where ptr becomes a void pointer

int *ptr=0;	Where ptr becomes a null pointer
int (*ptr)(int (*)[3], int (*)void))	Where ptr becomes a Complex pointer

2. Develop a C code to perform the following operation for a given text file

Replace the first 10 characters of the file to the digit 3

Replace the next 5 characters of the file to the character *

Replace the next 10 characters of the file with the digit 5

(Please note that, with -out using a second file, you have to rewrite the content in the same file itself)

Hint: sample.txt file content is AAAAAAAAAABBBCCCCCCCCCC before modification.

Content is 3333333333***5555555555 after modification

```
#include<stdio.h>                                        (9 marks)
#include<stdlib.h>
#include<ctype.h>
int main()
{
FILE *fp;
fp=fopen ("E:\\newfile.txt", "r+");   //w mode
char c;
int i;int j = 3;

for (i=0;i<=10;i++)
fputc('3',fp);    // fputc((int)'3',fp)     //putc('3',fp)        //fprintf(fp, "%d",j);
for (i=11;i<=15;i++)
fputc('*',fp);
for (i=16;i<=25;i++)
fputc('5',fp);
fclose(fp);
return 0;


}
```

(OR)

```
#include<stdio.h>                                          (10 marrks)

int main()

{

FILE *fp;

fp=fopen ("E:\\newfile12.txt", "w+");

char c;

int i;int j = 3;


while ((ftell(fp))<=10)

fputc('3',fp);    //fputc((int)'3',fp);//putc('3',fp)//fprintf(fp, "%d",j);

fseek(fp,10,0);


while (ftell(fp)<=15)

fputc('*',fp);    //fputc((int)'3',fp);//putc('3',fp)//fprintf(fp, "%d",j);

fseek(fp,15,0);


while (ftell(fp)<=25)

fputc('5',fp);    //fputc((int)'3',fp);//putc('3',fp)//fprintf(fp, "%d",j);

fclose(fp);

return 0;




}
```

3. Give the macro expansion of the following statement

```
#define minimum(A, B)  ((A) < (B) ? (A) : (B))
 x = minimum (a, b);          →  x = ((a) < (b) ? (a) : (b));
 y = minimum (1, 2);          →  y = ((1) < (2) ? (1) : (2));
 z = minimum (a + 28, *ptr);    →  z = ((a + 28) < (*ptr) ? (a + 28): (*ptr));
```

xyz = min(min(a,b), c) -→

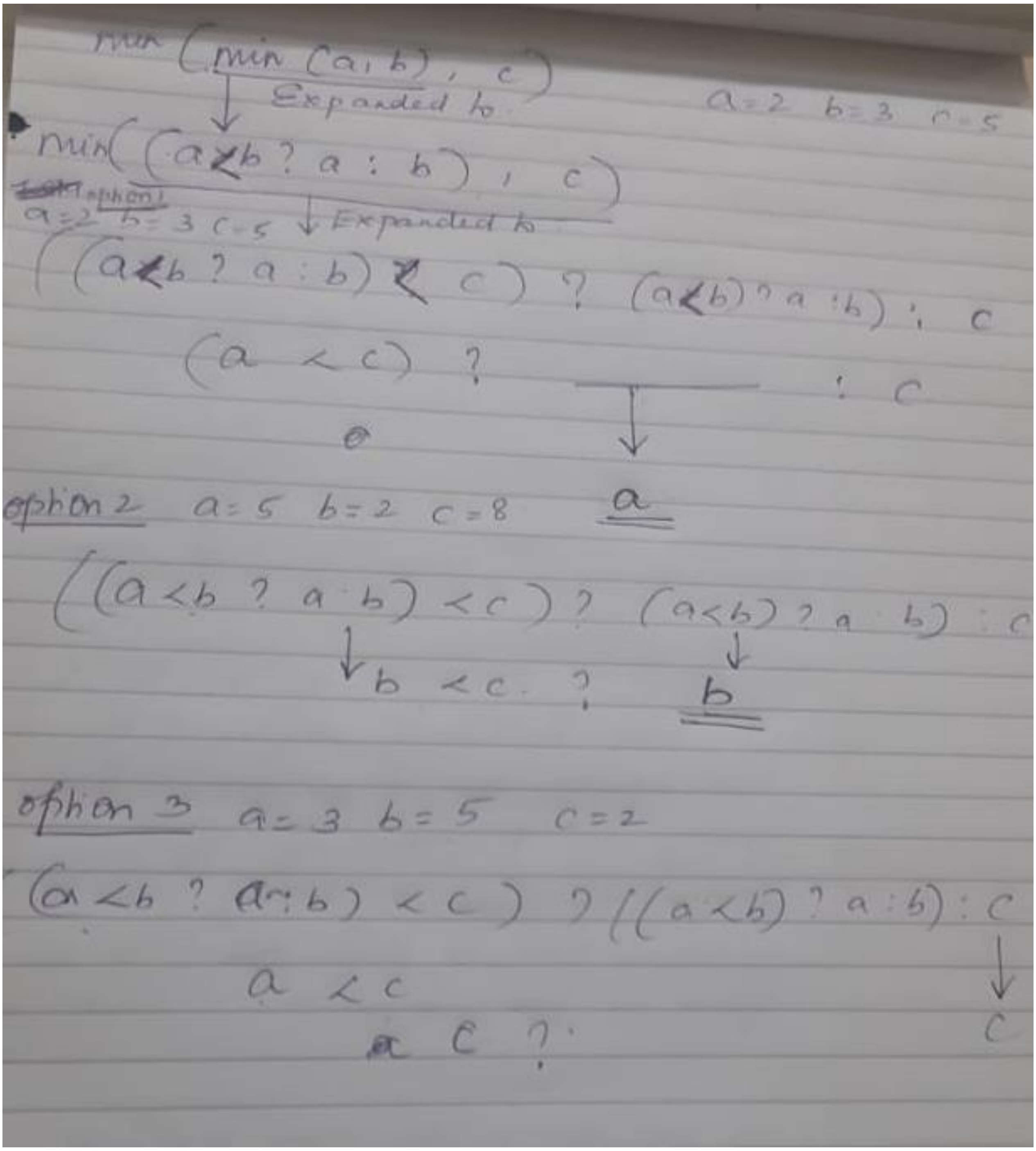
For example, `min (min (a, b), c)` is first expanded to

```
  min (((a) < (b) ? (a) : (b)), (c))
```

and then to

```
((((a) < (b) ? (a) : (b))) < (c)
 ? (((a) < (b) ? (a) : (b)))
 : (c))
```

Explanation of expanding the macro for the iv option of the question” xyz = min(min(a,b), c)



```
min (min (a, b), c)
    ↓ Expanded to                          a=2 b=3 c=5
min( (a<b ? a : b) , c)
~~1st~~ option1
a=2 b=3 c=5 ↓ Expanded to
((a<b ? a : b) < c) ? (a<b) ? a : b) : c
   (a < c) ?                         : c
                          ↓
                          a
option 2  a=5 b=2 c=8
((a<b ? a b) < c) ? (a<b) ? a b) : c
        ↓b < c. ?         ↓
                          b
option 3  a=3 b=5 c=2
(a<b ? a : b) < c) ? ((a<b) ? a : b) : c
      a < c                              ↓
        a c ?                            c
```