WEEK-END ASSIGNMENT-07

Pointers, m-D array and Dynamic Memory Allocation

Operating Systems Workshop (CSE 3541)

Problem Statement:

Experiment with pointers, 1-D & m-D array processing through pointers and dynamic memory allocation in C.

Assignment Objectives:

To learn how to manipulate arrays using pointers and to learn **malloc**, **mcalloc**, **realloc** & **free** to allocate and free dynamic memory.

Instruction to Students (If any):

Students are required to write his/her own program by avoiding any kind of copy from any sources. Additionally, They must be able to realise the outcome of that question in relevant to systems programming. You may use additional pages on requirement.

Programming/ Output Based Questions:

1. Consider the following ANSI C program;

```
#include<stdio.h>
int main() {
   int arr[4][5];
   int i,j;
   for(i=0;i<4;i++) {
      for(j=0;j<5;j++) {
        arr[i][j]=10*i+j;
      }
   }
   printf("%d\n",arr[2][4]);
   printf("%d\n",*(*(arr+2)+4));
   return 0;
}</pre>
```

2. Consider the following ANSI C program;

```
#include<stdio.h>
int main() {
    int arr[4][5];
    int i,j;
    for(i=0;i<4;i++) {
        for(j=0;j<5;j++) {
            arr[i][j]=10*i+j;
        }
    printf("%d\n",*(arr[1]+9));
    return 0;
}</pre>
```

What is the output of the above program?

Output with explanation						
(A) 14	(C) 24					
(B) 20	(D) 30					
(A) 24	(C) 14					
(B) 30	(D) 20					
Ans: 24 24						

[GATE 2021]

What is the output of the above program?

Output with explanation									
(A) 14	(C) 24								
(B) 20	(D) 30								
Ans: 24									

3. What is printed by the following ANSI C program?

[GATE 2022]

```
#include<stdio.h>
int main(void)
{
  int x = 1, z[2] = {10, 11};
  int *p = NULL;
  p = &x;
  *p = 10;
  p = &z[1];
  *(&z[0] + 1) += 3;
  printf("%d, %d, %d\n", x, z[0], z[1]);
  return 0;
}
```

```
Output with explanation

(A) 1, 10, 11
(B) 1, 10, 14
(C) 10, 14, 11
(D) 10, 10, 14

Ans: D) 10, 10, 14
```

4. What is printed by the following ANSI C program?

[GATE 2022]

```
#include<stdio.h>
int main(int argc, char *argv[])
{
  int a[3][3][3] =
  {{1, 2, 3, 4, 5, 6, 7, 8, 9},
  {10, 11, 12, 13, 14, 15, 16, 17, 18},
  {19, 20, 21, 22, 23, 24, 25, 26, 27}};
  int i = 0, j = 0, k = 0;
  for( i = 0; i < 3; i++ ){
    for(k = 0; k < 3; k++ )
      printf("%d ", a[i][j][k]);
    printf("\n");
}
return 0;
}</pre>
```

```
Output V

1 2 3

10 11 12

19 20 21
```

5. What is printed by the following ANSI C program?

[GATE 2022]

```
#include<stdio.h>
int main(int argc, char *argv[])
{
  int a[3][3][3] =
   {{1, 2, 3, 4, 5, 6, 7, 8, 9},
   {10, 11, 12, 13, 14, 15, 16, 17, 18},
   {19, 20, 21, 22, 23, 24, 25, 26, 27}};
  int i = 0, j = 0, k = 0;
  for( i = 0; i < 3; i++ ){
    for(k = 0; k < 3; k++ )
        printf("%d ", a[i][j][k]);
    printf("\n");
}
return 0;
}</pre>
```

Output▼											
ASCII encoding for relevant characters is given below											
A	В	С		Z							
65	66	67		90							
a	b	С		Z	*	+	-				
97	98	99		122	42	43	45				
(A) z K S (C) *-+											
(B) 122 75 83					(D) P x +						
Ans: 1 2 3 10 11 12 19 20 21											

6. Consider the program below;

[GATE 2009]

```
#include<stdio.h>
int fun(int n, int *f_p){
   int t,f;
   if(n<=1) {
      *f_p=1;
      return 1;
   }
   t=fun(n-1, f_p);
   f=t+ *f_p;
   *f_p=t;
   return f;
}</pre>
```

7. Consider the following C program

8. Consider the following C function;

```
int tob(int b, int *arr){
  int i;
  for(i=0;b>0;i++){
    if(b%2)
        arr[i]=1;
    else
        arr[i]=0;
    b=b/2;
}
return(i);
```

The value returned by **pp (3, 4)** is _____

```
int main() {
  int x=15;
  printf("%d\n", fun(5,&x));
  return 0;
}
```

The value printed is

```
Output V

(A) 6 (C) 14

(B) 8 (D) 15

Ans: B) 8
```

[GATE 2020]

The output of the program is

```
Output▼

Ans: 19
```

[GATE 2020]

```
int pp(int a, int b) {
   int arr[20];
   int i,tot=1,ex,len;
   ex=a;
   len=tob(b,arr);
   for(i=0;i<len;i++) {
      if(arr[i]==1) {
            tot=tot*ex;
      ex=ex*ex;
      return(tot);
}</pre>
```

```
Write the execution pattern and final output ▼

(i) len: 3

(ii) arr content: {0, 0, 1}

(iii) tot: 3

(iv) ex: 9

Finally, pp (3, 4): 3
```

9. Write the output of the following program;

```
#include<stdio.h>
void fun(int *,int *);
int main()
{
  int i=5,j=5;
  fun(&i,&j);
  printf("%d %d\n",i,j);
  return 0;
}
```

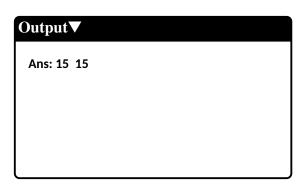
```
void fun(int *p,int *q) {
   p=q;
   *q=10;
}
```

```
Output V

Ans: 5 10
```

10. Find the output and different types of pointer involved in the code snippet;

```
int main() {
   int *p=NULL;
   p=(int *)malloc(sizeof(int));
   *p=10;
   free(p);
   int *q;
   q=(int *)malloc(sizeof(int));
   *q=15;
   printf("%d %d\n",*p,*q);
   return 0;
}
```



11. State the output of the following program. Assume the address of p is 1000 and q is 2000.

```
#include<stdio.h>
#include<stdlib,h>
void fun(int **q);
int main() {
  int *p=(int *)malloc(sizeof(int));
  *p=55;
  fun(&p);
  printf("%d %p\n",*p,p);
  return 0;
}
```

```
void fun(int **q) {
   int r=20;
   **q=r;
   printf("%p\n",*q);
}
```

```
Output V

0x20952a0
20 0x20952a0
```

12. Write the output of the code snippet by observing the co-relation of pointer manipulation in 2-D array.

```
int main(){
 int n=4, m=3;
 int a[n][m];
 int (*p)[m]=a;
 p=p+1;
  (*)p[2]=100;
 n=p-a;
 printf("%d\n",n);
                           /*---(A) */
 printf("%d\n",(*p)[2]); /*---(B) */
 printf("%d\n", *((*p)+2));/*---(C) */
 printf("%d\n", *(a[1]+2));/*---(D) */
 printf("%d\n", *(*p+2)); /*---(E) */
 printf("%d\n", *(p[0]+2));/*---(F) */
 return 0;
}
```

```
Output▼

(A) 1

(B) 0

(C) 0

(D) 0

(E) 0

(F) 0
```

13. Select the output of the following program.

```
int main() {
  int a[][3]={4,5,6,7,8,9,1,2,3};
  printf("%d,", *a[2]);
  printf("%d,", a[2][0]);
  printf("%d ", **(a+1+('b'-'a')));
  return 0;
}
```

ASCII value of a=97 and b=98

```
Output (A) 1024,1,1 (C) 1024,2,1024
(B) 1,1,1 (D) None of these
Ans: B
```

14. Select the desire output of the following code snippet with reason;

Output with reason▼

- (A) Unexpected behavoir
- (C) 30
- (B) Address of sum
- (D) None of these

Ans: A

15. Select the desire output of the following code snippet with reason;

Output with reason ▼

- (A) Unexpected behavoir
- (C) 30
- (B) Address of sum
- (D) None of these

Ans: C

16. Select the output of the following program..

```
int main() {
  int *ptr;
  ptr=(int *)realloc(NULL, sizeof(int));
  *ptr=100;
  printf("%d\n",*ptr);
  return 0;
}
```

Output ▼

100

17. Write the output of the following program.

```
1 int main(){int *ptr;
2  ptr=(int *)calloc(1,sizeof(int));
3  *ptr=100;
4  printf("%d\n",*ptr);
5  ptr=(int *)realloc(ptr,0);
6  ptr=NULL;
7  printf("%p\n",ptr);
8  return 0;}
```

Output ▼

Output at line-4: 100

Output at line-7: (nil)

Line number-6 can be treated as like **free()** to deallocate memory-**Y|N**.

int main() {int b=65; void p=b;printf("%d",p);
 return 0;}

Observation ▼

ERROR!

19. Select the output of the following program.

```
int main() {
  int b=65;
  void *p=&b;
  int *j=(int *)p;
  char *ch=(char *)p;
  printf("%d %c\n",*j,*ch);
  return 0;
}
```

```
Output (A) 65 65 (C) Compile time error

(B) 65 A (D) Run time error
Ans: B
```

20. Write the output of the code snippet. Also show the stack and heap memory for this application.

```
int main() {int i;
  int *p=(int *)malloc(sizeof(int));
  *p=100;
  p=(int *)malloc(5*sizeof(int));
  for(i=0;i<5;i++) {
      scanf("%d",p+i); /* 10,20,30,40,50 */
  }
  for(i=0;i<5;i++) {
    printf("%d...%d\n",p[i],*(p+i));
  }
  return 0;}</pre>
```

```
Output▼

2
14
14...14
25
25...25
14
14...14
14
14
14
14
```

21. Write the output of the code snippet. Also show the stack and heap memory for this application.

```
int main() {
int i,*p,*rp;
p=(int *)malloc(5*sizeof(int));
for(i=0;i<5;i++)
    scanf("%d",p+i); /* 10,20,30,40,50 */
rp=(int *)realloc(p,10*sizeof(int));
for(i=5;i<10;i++)
    scanf("%d",rp+i);/* 9,8,6,5,4 */
for(i=0;i<10;i++) {
    printf("%d...%d\n",rp[i],*(rp+i));
}
return 0;}</pre>
```

```
Output ▼
             52...2
 45
 4
             45...45
 7
             7...7
 8
             8...8
             44...44
 44
             57...57
 57
 5
             5...5
 58
             58...58
             69...69
 69
            8...8
 58
```

22. Which of the given statements about the following code snippet is/are correct?

```
void fun() {
    int *q=(int *)malloc(sizeof(int));
    *q=20;
}
int main() {
    int *p;
    int *r=NULL;
    fun();
    return 0;
}
```

```
(i) p is a wild pointer
(ii) r is a NULL pointer
(iii) q is dangling pointer
(iv) p is dangling pointer
(v) fun() is making memory leak
```

```
Output 2

dash: 2: 2: not found

dash: 3: 1: not found

dash: 4: 3: not found

5
```

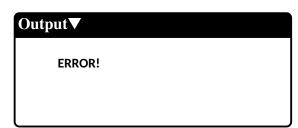
23. Which of the following statements are true?.

```
(1) (void *)0 is a void pointer
(2) (void *)0 is a NULL pointer
(3) int *p=(int *)0; p is a NULL pointer
(4) a[i]==i[a]
(5) a[i][j]== *(*(a+i)+j)
```

Output V

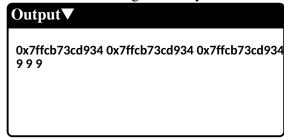
Statements 1 through 5 are all true.

24. Check the error or output of the following program?

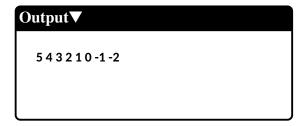


25. Write the output of the following program? Assume that the base address of a given array **a** is 1000?

```
int main() {
int a[3][3]={4,5,6,7,8,9,1,2,3};
printf("%p %p %p\n",a[1]+2,*(a+1)+2,&a[1][2])
   ;
printf("%d %d %d\n",*(a[1]+2),*(*(a+1)+2), a
   [1][2]);
return 0;
}
```



26. State the output of the code.



27. Write the output of the given code snippet.



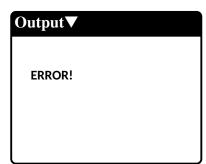
28. Write the output of the given code snippet that uses pointer to function or function pointer.

```
int fun(int x,int y) {
  int z=x+y+x*y;
  return z;
}
```

```
#include<stdio.h>
int main() {
  int (*fun_ptr)(int,int);
  fun_ptr=fun;
  int x=fun_ptr(34,56);
  printf("%d\n",x);
  return 0;
}
```



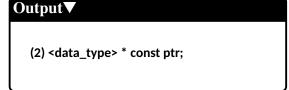
29. Mention the output of the following code snippet. [Array of pointers to function returning int type].



30. Find out the correct syntal(s) for making a constant pointer (i.e. The value of the pointer is constant and pointer cannot be modified).

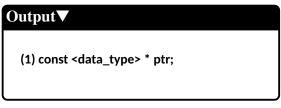
```
(1) const <data_type> * ptr;
(2) <data_type> * const ptr;
(3) <dat_type> const *ptr;
(4) <data type> const * const fun ptr
```

(5) None of these



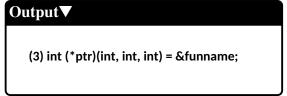
31. Find out the correct syntal(s) for a pointer to constant (i.e. The pointer cannot able to change the value of the variable/array that it points).

```
(1) const <data_type> * ptr;
(2) <data_type> * const ptr;
(3) <dat_type> const *ptr;
(4) <data_type> const * const fun_ptr
(5) None of these
```



32. Select the correct way of declaring and initializing pointer to function (i.e. function pointer).

```
(1) int (*ptr)(int,int,int)=funname;
(2) int *ptr(int,int,int)=funname;
(3) int (*ptr)(int,int,int)=&funname;
(4) (int *) ptr(int,int,int)=funname;
(5) None of these
```



33. Find the output of the code snippet.

```
int main() {
  int a[][2][4]={5,6,7,8,9,11,12,1};
  printf("%d\n",*(*(a+0)+1)+2));
  return 0;
}
```

```
Output▼
12
```

34. Describe the output for the following code snippet.

```
void fun(int arr[][3]) {
   printf("%d\n",*(*(arr+2)+1));
   printf("%p\n",(*arr)+2);
   printf("%p\n",&arr[0][2]);
   printf("%d\n",*(((*arr)+1)+1));
}
int main() {
   int a[][3]={5,6,7,8,9,4,3,2,1};
   fun(a);
   return 0;
}
```

```
2
0x7fff0be96838
0x7fff0be96838
7
```

35*. Explain the below declaration(s).

Output ▼

int process(int (*pf)(int a, int b));

This declares a function named process that takes a single argument, which is a pointer to a function. The function pointer pf should point to a function that takes two int parameters (int a and int b) and returns an int.

int (*fun(int, void (*ptr)()))();

This declares a function named fun that takes two arguments: an int and a pointer to a function that takes no arguments (void (*ptr)()) and returns void. The function fun itself returns a pointer to a function that takes no arguments and returns an int.

int *(*p)(int (*a)[]);

This declares a pointer p to a function that takes a pointer to an array of integers (int (*a)[]) as an argument and returns a pointer to an integer (int *).

int (*p)[10];

This declares a pointer p to an array of 10 integers. It does not initialize the pointer.

float *p[20];

This declares an array p of 20 pointers to float. Each element of the array is a pointer to a float.

int p(char *a)

This declares a function named p that takes a single argument, which is a pointer to a character (char *a), and returns an int.

int (*p(char * a))[10];

This declares a function named p that takes a single argument, which is a pointer to a character (char *a), and returns a pointer to an array of 10 integers.

int * (*p[10]) (char *a);

This declares an array p of 10 pointers to functions. Each function takes a single argument, which is a pointer to a character (char *a), and returns a pointer to an integer (int *).