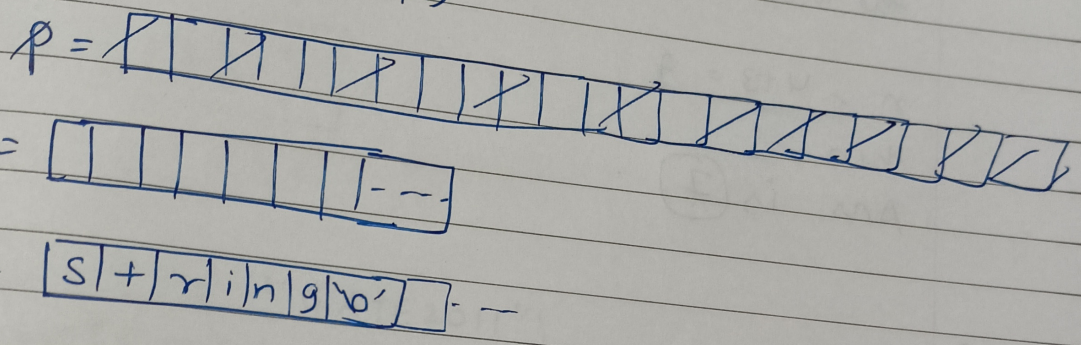
tutorial-1

Shashant mudgal
211B287

Q-1)

```
char p[20];
char *s = "string";
int length = strlen(s);
int i;
for(i=0; i<length; i++)
    p[i] = s[length-i];
Printf("%s", p);
```

Sol)

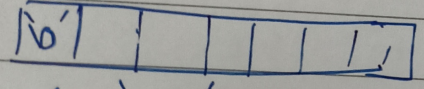


length = 6

So now

p[0] = s[6]

So



b/c of '\0' at indix 0 compiler end that program and we will get no output

no output

Q-3)

```
intf(int n){
 Static int i=1;
 if (n>=5)
   return n;
 n = n+i;
 i++;
 intf(n);
```

Soln)

at $n=1$, $i=1$

$n = 1+1 = 2$

now $n=2$, $i=2$

$n = 2+2 = 4$

&

now $n=4$, $i=3$

$n = 4+3 = 7$

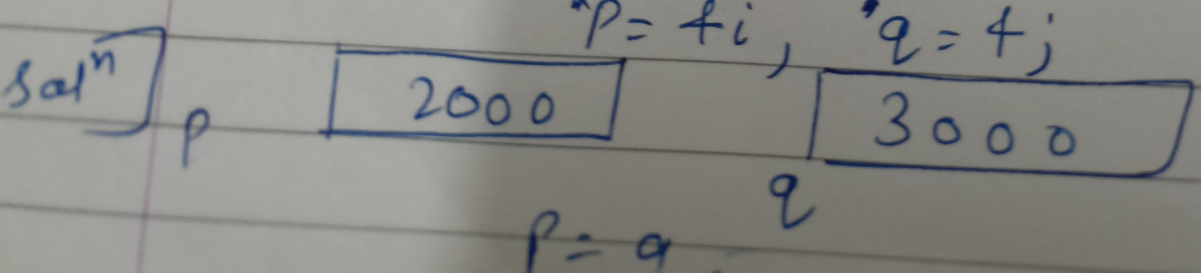
Then

Ans is (7).

4)

```
#include <stdio.h>
void f(int *p, int *q)
{
  p=q;
 *p=2;
}
int i=0, j=1;
int main()
{
 f(&i, &j);
 printf("\n%d %d\n", i, j);
 getchar();
 return 0;
}
```

Soln



2115287 modgal

p [3000] , q [3000] ; *p = fj, *q = fj

i [0] j [1]

now

*p = 2

and

*j = 2

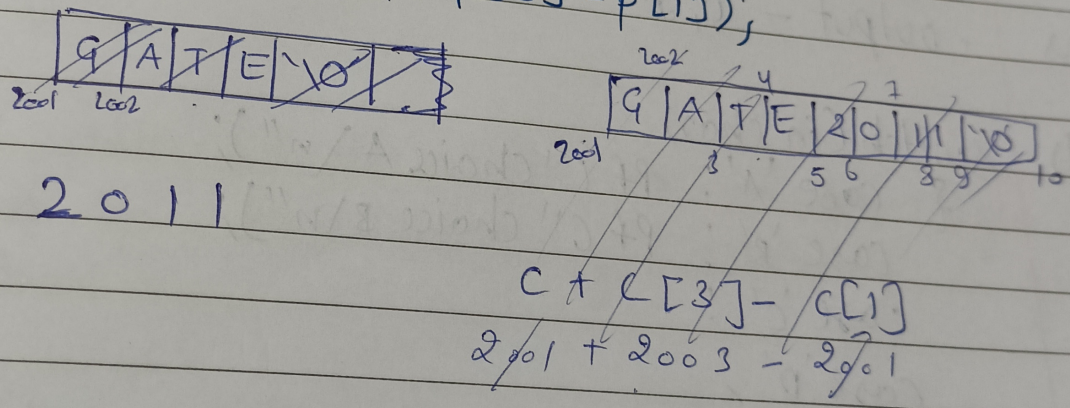
i = 0

then

ans [(0, 2)]

5)

```
char c[] = "GATE2011";
char *p = c;
printf("%d", p+p[3]-p[1]);
```



soln) 2011

6) unsigned long int fun (unsigned long int n){

```
    unsigned long int i, j, sum = 0
    for (i=n ; i>1 ; i=i/2){
        j++
    }
    for (j>1 ; j=j/2){
        sum++
    }
}
return sum;
```

Shashank mudgal
21IB287

ans output for fun (2^40) input is - 34

7) #include <stdlib.h>

8)
```
#include <stdio.h>
struct ournode{
    char x,y,z;
};
int main()
{ struct ournode p = {'1', '0', 'a'+2};
  struct ournode *q = &p;
  printf("%c %c %c", *(char*)q+1), *((cha
  return 0;
}
```

Ans output - 0, c

9)
```
switch (in char)
{ case 'A': Pf ("choice A \n");
  case 'B': Pf ("choice B\n");
  case 'C':
  case 'D';
  case 'E';
  default: pf ("No choice");
}
```

Ans) output - choice A
choice B No choice

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

Mudgal
211B287

```
#include <stdlib.h>
#include <string.h>
int main()
{ char *c = "GATECSIT 2017";
  char *p = c;
  printf("%d", (int)strlen(c+2[p] - c[p]-1));
  return 0;
}
```

Ans:- The output of the program is ____

Q-1)
```
struct {
    short s[5];
    union {
        float y;
        long z;
    } u;
} t;
```

Ans: Memory allocated to variable t will be 18bytes. T
description of the memory allocation is giv
as follows -

5*2 bytes = 10bytes → short
8bytes → long
∴ 10+8 = 18bytes

Tutorial-2

Q) Mention problem with structural programming and how oop's resolve them?

aus. structured programming approach has generally 4 problems which are efficiently resolved by using oop paradigm. these are-

(i) Not suitable for real world modelling-

structured programming lang. usually focus on only one real world concept ie data functions and not its attributes. whereas oop's supports real world modelling by supporting both data attribs as well as its functions using concepts like inheritance and polymorphism.

(ii) Not suitable for large and Complex softwares-

structural programming has a major flaw of restricted reusability of pre-written code: functions act as major concepts which to some extent add this functionality to structured programming. so code for softwares become quite complex to debug and maintain. oop's solve this problem by greatly increasing code reusability with inheritance concepts.

iii) unrestricted Access to global data.

Any unintentional changes to globally available data in the program may render junk results. to avoid this oop concept of data

21R287

hiding and its abstractions.

(iv) Dependency of function.

Being a bit modular in nature. function interdependency is a major flaw in structural programming approach. Any minimal changes to a single function connected to a chain of vital programm functions may disrupt the programm or render it useless entirely.

Q-2) write's oop's concepts with Examples.

ans object oriented programming Concepts include -

a) Encapsulation - It is the binding of data attributes and its functions in a singe unit. It reduces programm complexity and increases code reusability It is also known as wrapping up of data eg: use of class is a valuable eg of using encapsulation.

b) Inheritance - It is the concept used in oop's to improve code reusability by reimplementing previously written code's characteristics in some other code.

Eg:- Inheritance in classes

c) polymorphism - It is the concept of using same logic with diffrent condition; i.e testing same input with multipk euriroments eg: It is implemented using function overloading.

d) Data Abstraction – It is the concept which allows users to represent only useful information and hide the rest of it.

Eg: Header files used in oof supported lang. use data abstraction to hide actual code inside it.

e) Data hiding – It is used in oof's to prevent accidental / intentional data modification by hiding it from the user. It is implemented using concept of access specifiers.

Q-3) Describe dynamic memory allocation and also Explain memory allocation functions?.

ans :- Dynamic memory Allocation is the concept of initializing a variable by providing it memory during runtime of the programm. functions used for such allocation include –

(i) malloc() → It is used to allocate a check of memory as per user demand and to initialize each block by a garbage values. It is a generic func^n whose return type is void.

(ii) calloc() → It is the same as malloc, with the diffrence being that it initializes each memory block using zeroes. It takes two arguments i.e starting address & size of each block. It is also a

a generic func() of return type void.

(iii) realloc() → It is used to reallocate previously allocated memory as per new changes keeping original data in. in fact it is used to grow or shrink a memory block allocated ~~memory~~ ~~as for~~ to a variable.

(iv) free() → It is used to free the memory which is no longer in use in the programm. It is a garbage collector function triggered by user as per requirement of the programm.

Q-4) find the outcome

Q5
```
#include <stdio.h>
int main()
{ int k;
  int a[] = {1,2,3}
  int *b[3];
  int **c[3];
  int ***d[3];
  int ****e[3];
  int *****f[3];
for (k=0; k<3; k++)
        { b[k] = a+k;
          c[k] = b+k;
          d[k] = c+k;
          e[k] = d+k;
          f[k] = e+k;
```

Shashank mudgal
211B287

```
for (k=0; k<3; k++)
    { pf("%3d", *b[k]);
      pf("%3d", **c[k]);
      pf("%3d", ***d[k]);
    }
  return 0;
}
```

ans output : 111 222 333

(ii)

```
#include (stdio.h)
int main()
{ int a = 2, *p, **q;
  p = &a;
  q = &p;
  pf("%d %d %d", a, *p, **q);
```

ans output : 2 2 2

(iii)

```
#include <stdio.h>
void fun(int *ptr) {
    *ptr = 30;
}
int main()
{ int y = 20;
  fun(&y);
  pf("%d", y);
  return 0;
}
```

ans output : 30

Q-8) Find the errors.

K. Mudgal

21IB287

(i) int main()
{ int t[] = {1,2,3,4,5};
int *p, *q, r;
p=t, q=p[1], r=p[2];
printf("%d %d %d", *p, *q, *r); return 0; }

ans) Lines q=p[1] and r=p[2] are converting an integer ~~value~~ values to a pointer type variable without proper typecasting.

(ii) int main()
{ char *c;
float x=10;
c=fx;
printf("%d", fc);
return 0;
}

ans) Line d=fn is a long in the programm as c is a char type pointer whereas x is a floating point value; so c is incompatible with value of x; thus an error occurs while compiling the programm.

Q-6) find the outcome

#include <stdio.h>
int main(){
int *ptr;
int x;
ptr = &n;
*ptr=0;
printf("x = %d\n", n);
*ptr=3;

21IB287

```
printf("x = %d\n", x);
printf("*ptr = %d\n", *ptr);
(*ptr)++;
printf("x = %d\n", x);
printf("*ptr = %d\n", *ptr);
return 0;
}
```

ans) output
x = 0
*ptr = 0
n = 5
*ptr = 5
x = 6
*ptr = 6

c) Consider a compiler where int takes 4 byte, char take 1 byte and pointer takes 4 bytes. find the output?.

ans)

```
#include <stdio.h>
int main() {
int arri[] = {1,2,3};
int *ptri = arri;
char arrc[] = {1,2,3};
char *ptrc = arrc;
printf("size of arri[] = %d\n", size of (arri));
printf("size of ptri = %d", size of (ptri));
printf("size of arri[] = %d", size of (arrc));
printf("size of ptrc = %d", size of (ptrc));
return 0;
}
```

size of arri[] = 12 size of ptri = 4 size of arrc[] = 3
size of ptrc = 4

2118287

Q-8) what are the two ways to change the ranking variable to 45?.

```
struct video{
    char name[50];
    int ranking;
};
int main()
{ struct video cats = {"catvid", 53};
  struct video *ptr;
   ptr = &cats;
   return 0;
}
```

Ans) 1st method —

```
struct video{
    char name[50];
    int ranking;
};
int main() {
struct video cats = {"catvid", 53};
struct video *ptr;
 ptr = &cats;
cats.ranking = 45;
printf("%d\n", cats.ranking);
return 0;
}
```

2nd method - use of arrow operator :-

```
struct video{
    char name[50];
    int ranking;
};
int main()
```

Shashank Mudgal
211B287

```
{struct video cats = {"cat vid", 5};
 struct video *ptr;
  ptr = &cats;
  ptr -> ranking = 45;
  Pf("%d %d", ptr -> ranking);
  return 0;
}
```

Q-9) Assume that float take 4 bytes, predict the output of following programm:

```
# include <stdio.h>
  int main() {
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
    float *ptr1 = &arr[0];
    float *ptr2 = ptr1 + 3;
    Pf("%f", *ptr2);
    Pf("%d", ptr2 - ptr1);
    return 0;
  }
```

output - 90.500000 3.

Shashank mudgal
2111B287

```
{struct video cats = {"cat vid", 53};
 struct video *ptr;
  ptr = &cats;
  ptr -> ranking = 45;
  Pf("%d %d", ptr -> ranking);
  return 0;
 }
```

Q-9) Assume that float take 4 bytes, predict the output of following programm:

```
# include <stdio.h>
  int main() {
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
     float *ptr1 = &arr[0];
     float *ptr2 = ptr1 + 3;
     Pf("%f", *ptr2);
     Pf("%d", ptr2 - ptr1);
     return 0;
  }
```

output - 90.500000 3.

Shashank mudgal
21183287

```
{struct video cats = {"cat vid", 53};
 struct video *ptr;
  ptr = &cats;
  ptr -> ranking = 45;
  Pf("%d\n", ptr -> ranking);
  return 0;
 }
```

Q-9) Assume that float take 4 bytes, predict the output of following programm:

```
# include <stdio.h>
  int main() {
    float arr[5] = {12.5, 10.0, 13.5, 90.5, 0.5};
     float *ptr1 = &arr[0];
     float *ptr2 = ptr1 + 3;
     Pf("%f", *ptr2);
     Pf("%d", ptr2 - ptr1);
     return 0;
    }
```

output - 90.500000 3.

```
Void swap (int a, int b)
  {
     int temp = a;
      a = b;
      b = temp;
  }
  int main()
  {
   int a = 10, b = 20;
   swap (a, b);
   cout << a << b << Endl;   //10 //20
  }
```

Call by pointer.

```
void swap(int *a , int *b) {
    int temp = *a;
      *a = *b;
      *b = temp;
    }
int main() {
     int a=10, b=20;
      swap(&a, &b);
      cout<<a<<b<<endl   //20 //10
    }.
```