Vinit Choudhary
2218444

Tutorial-2

Ans 1) Problems with Structure programming:-

- same code repetition
- lack of encapsulation
- lack of information hiding

OOP method resolve them by Object, class, Abstraction, inheritance, Polymorphism and Encapsulation

Ans 2.) Class :- Collection of objects having a set of properties
class name :- name given to class.
Modifiers :- can be public or private
Inheritance :- Refers to inheriting properties of one class to another.
Encapsulation :- protection of data / wrapping of data
Abstraction :- displaying only essential information and hiding details.
Polymorphism :- Ability of a message to be displayed in more than one form

$\underbrace{\text{Poly}}_{\text{many}}\ \underbrace{\text{morphism}}_{\text{forms}}$

Ans 3.) It is the procedure in which the size of a data structure is changed during runtime.

Vinit Choudhary
221B444

Date: / /
Page No.

There are basically 4 functions:-

(1.) Malloc ( )
→ returns pointer of type void

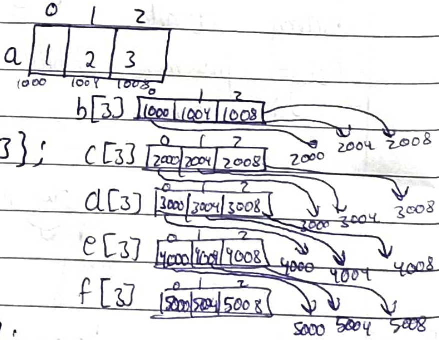
(2.) Calloc ( )
→ used to dynamically allocate the specified number of blocks of memory of specified type.

(3.) free ( )
→ It is used to de-allocate the memory

(4.) reallac ( )
→ It is used to change the previously allocated memory allocation.

Ans 4.) (i)

```
# 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++)
    {
```

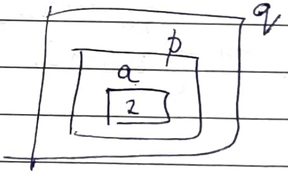


Vinit Choudhary
22184444

Date: / /
Page No.

```
                     for k=0           for k=1                 for k=2
  b[k] = a+k;        b[0] = a = 1000   b[1] = a+1              b[2] = 1008
                                        = 1000+1×4=1004
  c[k] = b+k;        c[0] = b = 2000   c[1] = 2004             c[2] = 2008
  d[k] = c+k;        d[0] = c = 3000   d[1] = 3004             d[2] = 3008
  e[k] = d+k;        e[0] = d = 4000   e[1] = 4004             e[2] = 4008
  f[k] = c+k;        f[0] = e = 5000   f[1] = 5004             f[2] = 5008.
}
for (k=0; k<3; k++)                    for k=0       for k=1   for k=2
{
    printf("%3d", *b[k]);    -> *b[0] = 1         2         3
    printf("%3d", **c[k]);   -> **c[0] = 1        2         3
    printf("%3d", ***d[k]);  -> ***d[0] = 1       2         3
}
return 0;
}
```

Output :-
1 1 1 2 2 2 3 3 3



```
(ii) # include <stdio.h>
  int main()
  {
    int a=2, *p, **q;
    p = &a;
    q = &p;
    printf("%d %d %d", a, *p, **q);      2   2   2
    return 0;
  }
```

Output :- 2 2 2

```
(iii) # include <stdio.h>
  void fun(int *ptr)
  {
```

Vinit Chaudhary
2218444

Date: / /
Page No.

```
    * ptr=30;
}
int main()
{
    int y=20;
    fun(&y);
    printf("%d", y);
    return 0;
}
```

*&y = 30

∴ y=30

30

Output :- 30

Ans5) (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;
}
```

Error :- Segmentation fault

(ii)

```
int main()
{
    Char * c;
    float x=10;
    c= &x;
    printf("%d", *c);
    return 0;
}
```

integer

char pointer

Vinit Choudhary
221B444

Ans 6)
```
# include <stdio.h>
int main()
{
    int * ptr;
    int x;
    ptr = & x;
    * ptr = 0;  → x=0
    printf ("x = %d\n", x);
    printf ("* ptr = %d\n", *ptr);
    * ptr += 5;  → x=5
    printf ("x = % d\n", x);
    printf ("* ptr = %d\n", * ptr);
    (* ptr) ++;  → x=6
    printf ("x = % d\n", x);
    printf (" * ptr = %d\n", *ptr);
    return 0;
}
```

Output:-
```
x=0
*ptr=0
x=5
* ptr=5
x=6
* ptr =6
```

Ans 7)
```
# include <stdio.h>
int main()
{
    int arri[] = {1, 2, 3};
    int * ptri = arri;
    char arrc[] = {1, 2, 3};
    char * ptrc = arrc;
    printf ("sizeof arri[] = % d", sizeof (arri));  → 4x3=12
    printf ("sizeof ptri = %d", sizeof(ptri));  → 4
    printf (" size of arrc[] = % d", sizeof (arrc));  → 1x3=3
```

int → 4 bytes
char → 1 byte
pointer → 4 bytes

Vinit Chaudhary
2218444

→ 4

```
printf("sizeof ptrc = %d", sizeof(ptrc));
return 0;
}
```

Output :-

```
sizeof arri[]=12  sizeof ptri =8
sizeof arrc[]=3   sizeof ptrc =8
```

Ans 8.)

```
struct video {
    char name[50];
    int ranking;
};
int main() {
    struct video cats ={"Cat Vid", 5};
    struct video * ptr;
    ptr = &cats;
    return 0;
}
```

→ ptr. ranking=45 or ptr→ranking=45

Ans 9.)

```
# include <stdio.h>
int main()
{
  float arr[5]={12.5, 10.0, 13.5, 90.5, 0.5};
  float * ptr1= & arr[0]; → 100
  float * ptr2= ptr1 +3; → 112
  printf ("%f", * ptr2); → prints 90.5
  printf ("%d", ptr2-ptr1); → prints 3
  return 0;
}
```

arr	12.5	10.0	13.5	90.5	0.5
	↑ 100	↑ 104	↑ 108	↑ 112	↑ 116

Output:- 90.5 3