LAB 13 (POINTERS)
ABHINAV ANAND ROLL-22052611 SEC-B16

```
//q1)Print the address of variables using the address operator.
#include<stdio.h>
void main(){     int age=10;     float sal=6500.25;
printf("Value of age=%d, Address of age=%u\n",age,&age);
printf("Value of sal=%f, Address of sal=%u\n",sal,&sal);
}
```

Output-

```
Value of age=10, Address of age=6422300
Value of sal=6500.250000, Address of sal=6422296
```

```
//q2)Program to dereference pointer variables.
#include<stdio.h>
void main(){     int a=10;     float b=6500.25;
int *p1=&a;     float *p2=&b;
printf("Value of p1 = Address of a = %u\n",p1);
printf("Value of p2 = Address of b = %u\n",p2);
printf("Address of p1 = %u\n",&p1);
printf("Address of p2 = %u\n",&p2);
printf("Value of a =%d %d %d \n",a,*p1,*(&a));
printf("Value of b =%f %f %f \n",b,*p2,*(&b));
}
```

Output-

```
Value of p2 = Address of b = 6422296
Address of p1 = 6422292
Address of p2 = 6422288
Value of a =10 10 10Value of b =6500.250000 6500.250000
6500.250000
```

```
//q3)Program to print the size of the pointer variable and size of value
dereferenced by that pointer.
#include<stdio.h>
void main(){
 int *ptr;     char b='c';     char *ptr2=&b;     int val = 10;     ptr =
&val;
 printf("The size of the pointer variable(int) is %u
bytes.\n",sizeof(ptr));
```

```
   printf("The size of the pointer variable(char) is %u
bytes.\n",sizeof(ptr2));     printf("The size of the value dereferenced
by the pointer(int) is %u bytes.\n",sizeof(*ptr));
   printf("The size of the value dereferenced by the pointer(char) is %u
bytes.\n",sizeof(*ptr2));
}
```

Output-

```
The size of the pointer variable(int) is 4 bytes.
The size of the pointer variable(char) is 4 bytes.
The size of the value dereferenced by the pointer(int) is
4 bytes.
The size of the value dereferenced by the pointer(char) is
1 bytes.
```

```
//q4)Program to show pointer arithmetic.
#include<stdio.h>
 void main(){     int a=5,*pi=&a;     char
b='x',*pc=&b;     float c=5.5,*pf=&c;
printf("Value of pi = Address of a = %u\n",pi);
printf("Value of pc = Address of b = %u\n",pc);
printf("Value of pf = Address of c = %u\n",pf);
pi++;     pc++;     pf++;
    printf("Now value of pi = %u\n",pi);
printf("Now value of pc = %u\n",pc);
printf("Now value of pf = %u\n",pf);
}
```

Output-

```
Value of pi = Address of a = 6422288
Value of pc = Address of b = 6422287
Value of pf = Address of c = 6422280
Now value of pi = 6422292
Now value of pc = 6422288
Now value of pf = 6422284
```

```
//q5)Program to understand pointer to pointer.
#include<stdio.h>
void main(){     int a=5;     int *pa;     int **ppa;
pa=&a;     ppa=&pa;
  printf("Address of a=%u\n", &a);
```

```
   printf("Value of pa= Address of a= %u\n",pa);
printf("Value of *pa= Value of a= %d\n",*pa);
printf("Address of pa=%u\n",&pa);
 printf("Value of ppa= Address of pa= %u\n",ppa);
printf("Value of *ppa= Value of pa= %u\n",*ppa);
printf("Value of **ppa= Value of a= %d\n",**ppa);
printf("Address of ppa= %u\n",&ppa);
}
```

Output-

```
Address of a=6422300
Value of pa= Address of a= 6422300
Value of *pa= Value of a= 5
Address of pa=6422296
Value of ppa= Address of pa= 6422296
Value of *ppa= Value of pa= 6422300
Value of **ppa= Value of a= 5
Address of ppa= 6422292
```

```
//q6)Program to print the value and address of elements of an array using
pointer notation. #include<stdio.h>
 void main(){
    int arr[5]={5,10,15,20,25};     int i;
for(i=0;i<5;i++){
 printf("Value of arr[%d]=%d\t",i,*(arr+i));
printf("Address of arr[%d]=%u\n",i,arr+i);     }
}
```

Output-

```
Value of arr[0]=5       Address of arr[0]=6422280
Value of arr[1]=10      Address of arr[1]=6422284
Value of arr[2]=15      Address of arr[2]=6422288
Value of arr[3]=20      Address of arr[3]=6422292
Value of arr[4]=25      Address of arr[4]=6422296
```

```
//q7)Program to understand the difference between pointer to an integer and
pointer to an array of integers.
#include<stdio.h>
void main(){
 int *p;     int (*ptr)[5];
 int arr[5];     p=arr;     ptr=arr;
printf("p =%u,ptr =%u\n",p,ptr);
p++;     ptr++;
printf("p =%u,ptr =%u\n",p,ptr);
}
```

LAB 13 (POINTERS)
ABHINAV ANAND ROLL-22052611 SEC-B16

Output-

```
p =6422276,ptr =6422276 p
=6422280,ptr =6422296
```

```
//q8)Program to dereference a pointer to an array.
#include<stdio.h> void
main(){
    int arr[5]={3,5,6,7,9};
int *p=arr;     int
(*ptr)[5]=arr;
    printf("p = %u, ptr = %u\n",p,ptr);
   printf("*p = %u, *ptr = %u\n",*p,*ptr);
printf("sizeof(p) = %u, sizeof(*p) = %u\n",sizeof(p),sizeof(*p));
printf("sizeof(ptr) = %u, sizeof(*ptr) = %u\n",sizeof(ptr),sizeof(*ptr));
}
```

Output-

```
p = 6422276, ptr = 6422276 *p = 3,
*ptr = 6422276 sizeof(p) = 4,
sizeof(*p) = 4 sizeof(ptr) = 4,
sizeof(*ptr) = 20
```

```
//q9)Add two numbers using call by reference.
#include<stdio.h>
void
main(){     int
a,b,sum;     a=4;
b=6;
func(a,b,&sum);
    printf("The sum is = %d",sum);
} func(int x,int y,int
*s){
    *s=x+y; }
```

Output-

```
The sum is = 10
```

```
//q10)Program to demonstrate how a 1D array is passed to a function
#include<stdio.h>
 void func(int a[]){     int i;
printf("Inside func() :");
```

LAB 13 (POINTERS)
ABHINAV ANAND ROLL-22052611 SEC-B16

```
for(i=0;i<5;i++){         a[i]
=a[i]+2;
printf("%d",a[i]);
    }
    printf("\n");
} void
main(){
    int i,arr[5]={3,6,2,7,1};
func(arr);
printf("Inside main() :");
for(i=0;i<5;i++){         prin
tf("%d", arr[i]);
    }
    printf("\n");
}
```

Output-

```
Inside func() :58493
Inside main() :58493
```

```
//q11)Add two matrices using the function.
#include <stdio.h>
void add(int a[][3], int b[][3], int result[][3], int rows, int cols)
 {     int i,j;     for(i=0;i<rows;i++){         for (j=0;j<cols;j++)
{              result[i][j]=a[i][j]+b[i][j];
        }
    } } int main(){     int a[3][3]={{1, 2,
3},{4, 5, 6},{7, 8, 9}};     int b[3][3]={{9, 8,
7},{6, 5, 4},{3, 2, 1}};     int result[3][3];
int i, j;     add(a, b, result, 3, 3);
printf("First Matrix :\n");
for(i=0;i<3;i++){         for(j=0;j<3;j++){
        printf("%d ",a[i][j]);
        }
        printf("\n");
    }     printf("\nSecond
Matrix :\n");     for(i=0;i<3;i++)
{         for(j=0;j<3;j++){
      printf("%d ",b[i][j]);
        }
        printf("\n");
    }
    printf("\nResult:\n");
for(i=0;i<3;i++)
{         for(j=0;j<3;j++){
            printf("%d ",result[i][j]);
        }
```

```
        printf("\n");
    }
return 0;
}
```

Output-

```
First Matrix :
1 2 3
4 5 6
7 8 9

Second Matrix :

9 8 7
6 5 4
3 2 1
 Result:
10 10 10
10 10 10
10 10 10
```