Pointers and Structures in C

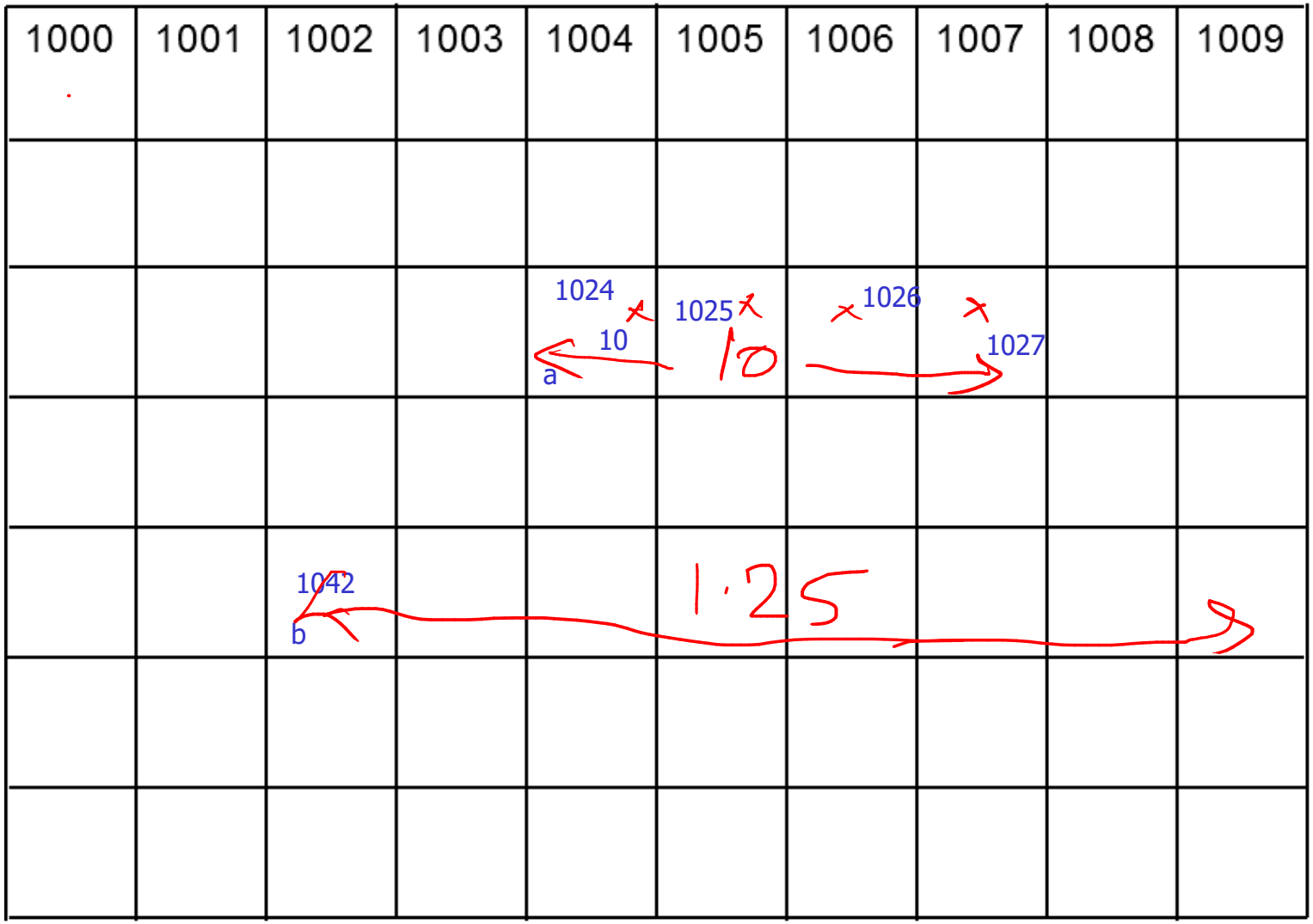
```c
#include <stdio.h>
int main()
{
  int a = 10;
  printf("a = %d\n", a);
  return 0;
}
```

int a;
a = 10;

4 bytes

float b = 1.25

8 bytes



```c
#include <stdio.h>
int main()
{
  int a = 10;
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  return 0;
}
```

Pointers

that stores the address of another variable

```
#include <stdio.h>
int main()
{
  int a = 10, *p;
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  p = &a;
  printf("p = %u\n", p);   //content in p
  printf("&p = %u\n", &p);//address of p
  printf("*p = %d\n", *p);//content pointed by p
  return 0;
}
```

int *p;
int* p;
int * p;

int a

1000	1001	1002	1003	1004	1005	1006	1007	1008	1009

Arrays

int a[1000];

```
#include <stdio.h>
int main()
{
  int a[] = {1, 2, 3, 4, 5};
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  return 0;
}
```

a[0], a[1]

1000	1001	1002	1003	1004	1005	1006	1007	1008	1009

Arrays

```
#include <stdio.h>
int main()
{
  int a[] = {1, 2, 3, 4, 5};
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("a[2] = %d\n", a[2]);
  return 0;
}
```

Arrays

```
#include <stdio.h>
int main()
{
  int a[] = {1, 2, 3, 4, 5};
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("a[2] = %d\n", a[2]);
  printf("&a[2] = %d\n", &a[2]);
  return 0;
}
```

Arrays

```
#include <stdio.h>
int main()
{
  int a[] = {1, 2, 3, 4, 5};
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("a[2] = %d\n", a[2]);
  printf("&a[2] = %d\n", &a[2]);
  printf("a+2 = %d\n", a+2);
  return 0;
}
```

Arrays

```
#include <stdio.h>
int main()
{
  int a[] = {1, 2, 3, 4, 5};
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("a[2] = %d\n", a[2]);
  printf("&a[2] = %d\n", &a[2]);
  printf("a+2 = %d\n", a+2);
  printf("*(a+2) = %d\n", *(a+2));
  return 0;
}
```

Dynamic Arrays

```
#include <stdio.h>
int main()
{
  int n;
  scanf("%d", &n);
  int a[n];
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("a[2] = %d\n", a[2]);
  printf("&a[2] = %d\n", &a[2]);
  printf("a+2 = %d\n", a+2);
  printf("*(a+2) = %d\n", *(a+2));
  return 0;
}
```

Dynamic Arrays

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int n, *a;
  scanf("%d", &n);
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("*a = %d\n", *a); // not safe

  a = (int*) malloc(n * sizeof(int));
  printf("a = %d\n", a);
  printf("&a = %u\n", &a);
  printf("*a = %d\n", *a);

  return 0;
}
```

1000	1001	1002	1003	1004	1005	1006	1007	1008	1009

Dynamic Arrays

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int n, *a;
   scanf("%d", &n);
   a = (int*) malloc(n * sizeof(int));

   printf("a = %d\n", a);
   printf("&a = %u\n", &a);
   printf("*a = %d\n", *a);

   printf("a+1 = %d\n", a+1);
//printf("&(a+1) = %u\n", &(a+1)); not valid - why?
   printf("*(a+1) = %d\n", *(a+1));

   return 0;
}
```

a a[0]
a+1 a[1]

a *a
a+1 *(a+1)
a[0] = *a
a[1] = *(a+1)

Passing Array to a function

```
#include <stdio.h>
#include <stdlib.h>

int sum(int* array, int n)
{
}

int main()
{
   int n, *a;
   scanf("%d", &n);
   a = (int*) malloc(n * sizeof(int));

   for(int i = 0; i < n; i++)
        scanf("%d", a+i); //&a[i]

   int s = sum(a, n);

   return 0;
}
```

int array[], int n

sum = 0;
for
sum = sum + array[i] ;// *(array+i)
retutn sum;

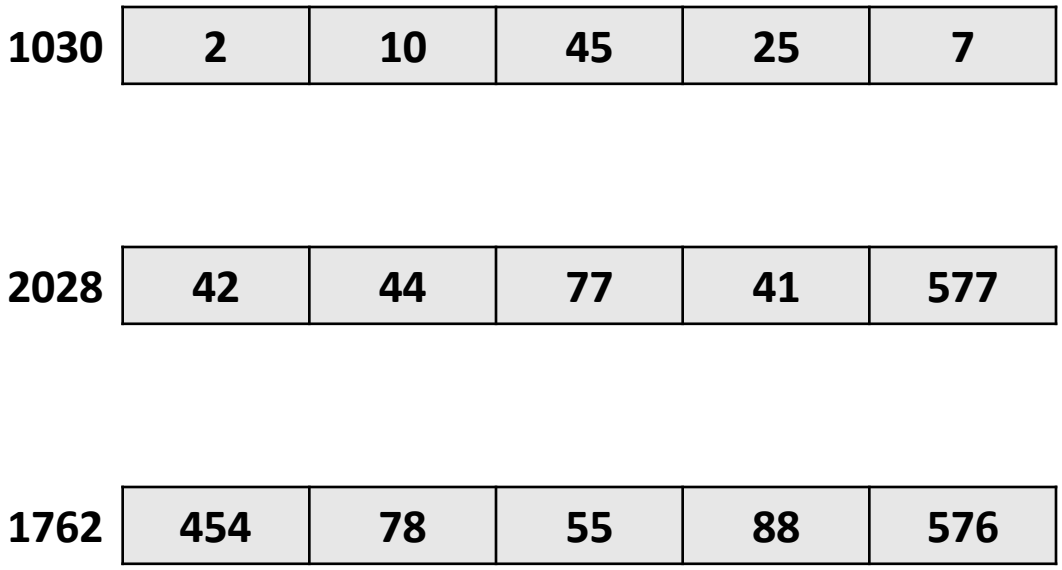
Return array from a function

```
#include <stdio.h>
#include <stdlib.h>

int* double(int* array, int n)
{
   int* d;
   d = (int*) malloc(n * sizeof(int));
    for()  d[i] = 2 * array[i]; // *(d+i) = 2 * *(array + i);
   return d;
}

int main()
{
   int n, *a;
   scanf("%d", &n);
   a = (int*) malloc(n * sizeof(int));

   for(int i = 0; i < n; i++)
        scanf("%d", a+i);

   int* s = double(a, n);
    //print s
   return 0;
}
```

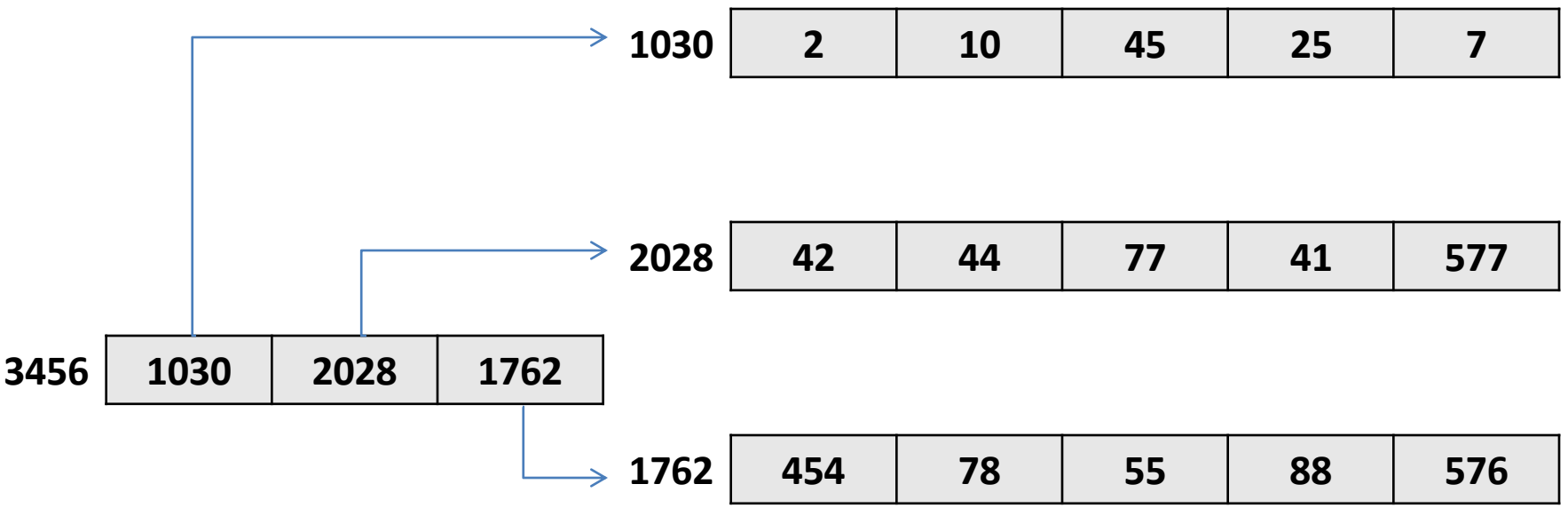
int[] double (int *array, int n) - wrong

double(a, n, d) - not recommended

Multiple Arrays



2D Arrays


1030
2
10
45
25
7
2028
42
44
77
41
577
3456
1030
2028
1762
1762
454
78
55
88
576

2D Arrays

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int n, **a;

   a = (int**) malloc(3 * sizeof(int*));

   printf("a = %d\n", a);
   printf("&a = %u\n", &a);
   printf("*a = %d\n", *a);

   printf("a+1 = %d\n", a+1);
//printf("&(a+1) = %u\n", &(a+1)); not valid - why?
   printf("*(a+1) = %d\n", *(a+1));

   return 0;
}
```

2D Arrays

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int n, i, **a;

  a = (int**) malloc(3 * sizeof(int*));

  for(i = 0; i < 3; i++)
     *(a+i) = (int*) malloc(5 * sizeof(int));
     //a[i] = (int*) malloc(5 * sizeof(int));

  return 0;
}
```

2D Arrays

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int n, i, **a;

  a = (int**) malloc(3 * sizeof(int*));

  for(i = 0; i < 3; i++)
  {
     *(a+i) = (int*) malloc(5 * sizeof(int));
     printf("*(a+i) = %u\n", *(a+i));
     printf("a[i] = %u\n", a[i]);
  }
  return 0;
}
```

Structures

```
struct employee
{
   char name[10];
   int salary;
   int work_per_day;
};
int main()
{
   int n;
   struct employee DB;

   printf("DB = %u\n", &DB);
   printf("DB = %u\n", &DB->name);
   printf("DB = %u\n", &DB->salary);
   printf("DB = %u\n", &DB->work_per_day);

   return 0;
}
```

Structures

```
struct employee
{
   char name[10];
   int salary;
   int work_per_day;
};
int main()
{
   int n;
   struct employee *DB;

   printf("DB = %u\n", &DB);
   printf("DB = %u\n", DB);
   printf("DB = %u\n", &DB->name);
   printf("DB = %u\n", &DB->salary);
   printf("DB = %u\n", &DB->work_per_day);

   return 0;
}
```

Array of Structures

```
struct employee
{
   char name[10];
   int salary;
   int work_per_day;
};

int main()
{
   int n;
   struct employee *DB;
   scanf("%d", &n);
   DB = (struct employee*) malloc(n * sizeof(struct
   employee));
   ……
   ……
}
```

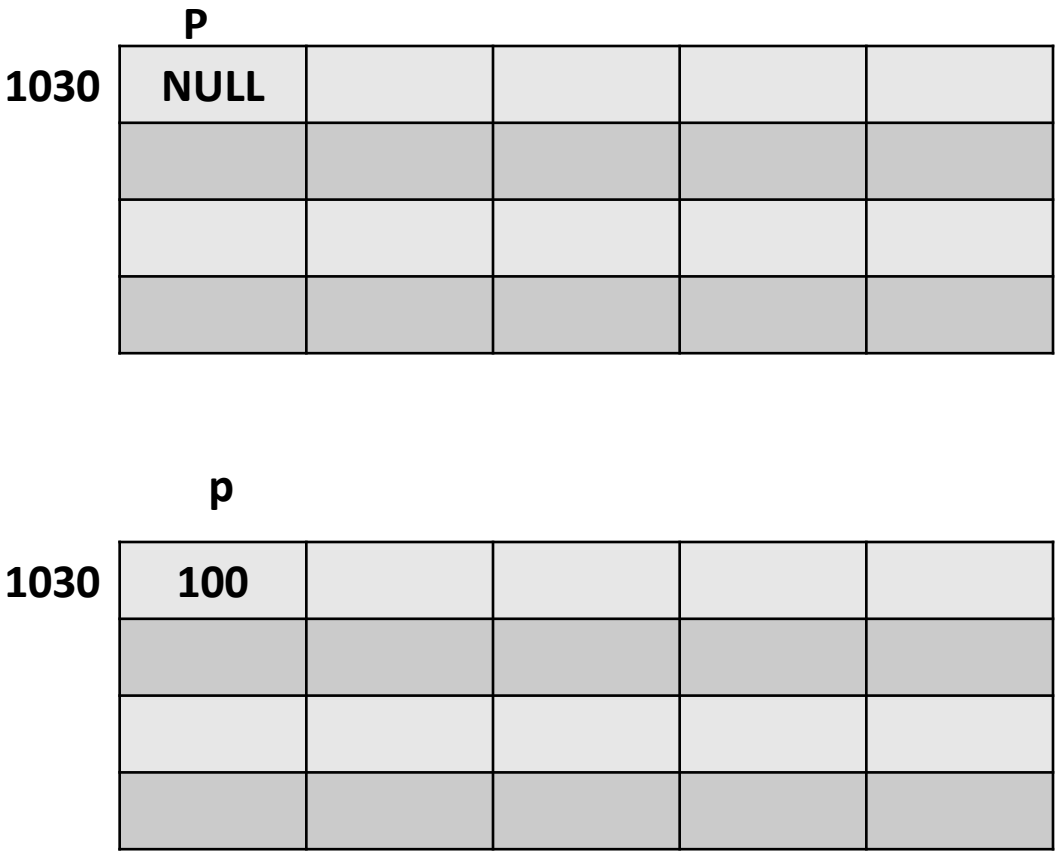
Segmentation fault??

```
#include <stdio.h>
int main()
{
  int *p = NULL;

  printf("p = %u\n", p);
  printf("&p = %u\n", &p);
  printf("*p = %d\n", *p);

  return 0;
}
```

Segmentation fault??

```
#include <stdio.h>
int main()
{
   int *p;

   printf("p = %u\n", p);
   printf("&p = %u\n", &p);
   printf("*p = %d\n", *p);

   p = 100;

   printf("p = %u\n", p);
   printf("&p = %u\n", &p);
   printf("*p = %d\n", *p);

   return 0;
}
```

Segmentation fault??



Segmentation fault is your fault,
not your computer's fault!!

How to get partial marks for implementation?

- Each function in the question carries some marks.
- First read and store the input, then print it.
- Code each function one by one and test its correctness.
- Don’t upload a code with compilation error.