

CS2263

FR02B

Gavin Hosford

3755030

January 30th, 2025

Lab #2

Q1)

It should not work as it is undefined behavior. Since we are trying to reference a point outside of the array. However, since my code is simple it is able to call it as there is no other information being stored alongside the code.

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

```
int main(){
```

```
    int array[5];
```

```
    array[0] = 10;
```

```
    array[1] = 58;
```

```
    array[2] = 25;
```

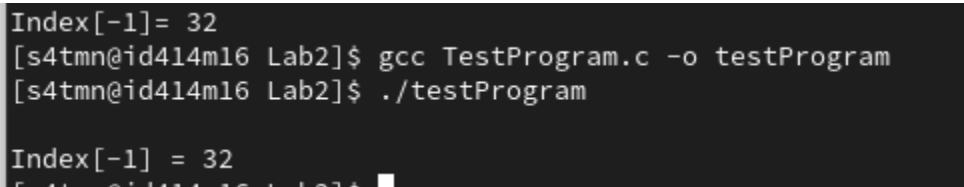
```
    array[3] = 9;
```

```
    array[4] = 27;
```

```
    array[-1] = 32;
```

```
    printf("\nIndex[-1] = %d\n", array[-1]);
```

```
}
```



```
Index[-1] = 32
[s4tmn@id414m16 Lab2]$ gcc TestProgram.c -o testProgram
[s4tmn@id414m16 Lab2]$ ./testProgram

Index[-1] = 32
[s4tmn@id414m16 Lab2]$
```

Q2)

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

```

int main(int argc, char **argv) { int arr1[] = {7, 2, 5, 3, 1, 6, -8, 16, 4};

char arr2[] = {'m', 'q', 'k', 'z', '%', '>'};

double arr3[] = {3.14, -2.718, 6.626, 0.529};

int len1 = sizeof(arr1) / sizeof(int);
int len2 = sizeof(arr2) / sizeof(char);
int len3 = sizeof(arr3) / sizeof(double);

printf("lengths = %d, %d, %d\n", len1, len2, len3);

int *iptr = arr1;
char *cptr = arr2;
double *dptr = arr3;

printf("values = %d, %c, %f\n", *iptr, *cptr, *dptr);
printf("address = %p, %p, %p\n", (void *)iptr, (void *)cptr, (void *)dptr);

iptr++; cptr++; dptr++;
printf("values = %d, %c, %f\n", *iptr, *cptr, *dptr);
printf("address = %p, %p, %p\n", (void *)iptr, (void *)cptr, (void *)dptr);

iptr++; cptr++; dptr++;
printf("values = %d, %c, %f\n", *iptr, *cptr, *dptr);
printf("address = %p, %p, %p\n", (void *)iptr, (void *)cptr, (void *)dptr);

iptr++; cptr++; dptr++;
printf("values = %d, %c, %f\n", *iptr, *cptr, *dptr);
printf("address = %p, %p, %p\n", (void *)iptr, (void *)cptr, (void *)dptr);

return EXIT_SUCCESS;

}

```

```
[s4tmn@id414m16 Lab2]$ ./p2
lengths = 9, 6, 4
values = 7, m, 3.140000
address = 0x7ffc7f2c3c00, 0x7ffc7f2c3bfa, 0x7ffc7f2c3bd0
values = 2, q, -2.718000
address = 0x7ffc7f2c3c04, 0x7ffc7f2c3bfb, 0x7ffc7f2c3bd8
values = 5, k, 6.626000
address = 0x7ffc7f2c3c08, 0x7ffc7f2c3bfc, 0x7ffc7f2c3be0
values = 3, z, 0.529000
address = 0x7ffc7f2c3c0c, 0x7ffc7f2c3bfd, 0x7ffc7f2c3be8p[s4tmn@id414m16
```

a) yes, they are incrementing by their respective bit amount.

b) The address for int changes by 4 bytes. The char only goes up by 1 byte. And the double increases by 8 bytes. Yes, they are the same for the same kind of variable. This is for uniform storage on the stack.

Q3)

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

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

```
//#include <String>
```

```
int main(){
```

```
    int arr1[] = {10, 11, 12, 13, 14, 15, 16};
```

```
    int * iptr = arr1;
```

```
    int n = (&arr1 + 1) - arr1;
```

```
    for(int i = 0; i < n; i++){
```

```
        printf("%d ", i);
```

```
        printf("%d ", arr1[i]);
```

```

    printf("%p", iptr);

    printf("%d \n", *iptr);

    *iptr++;

}

}

```

```

[s4tmn@id414m16 Lab2]$ gcc doubleArray.c -o double
[s4tmn@id414m16 Lab2]$ ./double
0 ,10 ,0x7ffd1c5875e0 ,10
1 ,11 ,0x7ffd1c5875e4 ,11
2 ,12 ,0x7ffd1c5875e8 ,12
3 ,13 ,0x7ffd1c5875ec ,13
4 ,14 ,0x7ffd1c5875f0 ,14
5 ,15 ,0x7ffd1c5875f4 ,15
6 ,16 ,0x7ffd1c5875f8 ,16

```

Q4)

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

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

```
int main(){
```

```
    int arrindex (int a[], int * p){
```

```
        int temp = (int)(p - a);
```

```
    return temp;
}

int arr[] = {10, 11, 12, 13, 14, 15, 16};
for (int i; i < sizeof(arr)/sizeof(arr[0]); i++){
    printf("%d %d \n", i, arrindex( arr, & arr[i]));
}
}
```

```
[s4tmn@id414m16 Lab2]$ gcc arrayInd.c -o index
[s4tmn@id414m16 Lab2]$ ./index
0 0
1 1
2 2
3 3
4 4
5 5
6 6
```