NAME : SAMIT DAS

ROLL : 2021ITB030

ASSIGNMENT : 4

1.WAP to display array elements with their addresses using the array name as a pointer.

CODE:

```
#include <stdio.h>
void display(int *arr, int n) {
    for(int i = 0; i < n; i++) {
        printf("%p → %d", arr+i, *(arr+i));
    }
    printf(“\n”);
}

int main() {
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    display(arr, 10);
    return 0;
}
```

OUTPUT:

```
samitdas:Downloads/ $ ./a.out
0x16b897470 -> 0
0x16b897474 -> 1
0x16b897478 -> 2
0x16b89747c -> 3
0x16b897480 -> 4
0x16b897484 -> 5
0x16b897488 -> 6
0x16b89748c -> 7
0x16b897490 -> 8
0x16b897494 -> 9
```

2. WAP to find the sum of an array's elements. Use the array name itself as a pointer.

CODE:

```
#include<stdio.h>


int getSum(int *arr, int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += *(arr+i);
    }
    return sum;
}


int main() {
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    int sum = getSum(arr, 10);
    printf("Array sum: %d\n", sum);
    return 0;
}
```

OUTPUT:

```
[samitdas:Downloads/ $ ./a.out
 Array sum: 45
```

3. WAP to store addresses of different elements of an array using an array of pointers.

CODE:

```
#include<stdio.h>


void pointerArray(int *arr, int n) {
    int* ptr[n];
    for(int i = 0; i < n; i++) {
        ptr[i] = arr + i;
    }
    for(int i = 0; i < n; i++) {
        printf("%p\n", ptr[i]);
    }
}


int main() {
    int arr[10] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    pointerArray(arr, 10);
    return 0;
}
```

OUTPUT:

```
samitdas:Downloads/ $ ./a.out
0x16b39f470
0x16b39f474
0x16b39f478
0x16b39f47c
0x16b39f480
0x16b39f484
0x16b39f488
0x16b39f48c
0x16b39f490
0x16b39f494
```

4. WAP to display the address of a user-defined function.

CODE:

```
#include<stdio.h>

int getSum(int *arr, int n) {
    int sum = 0;
    for(int i = 0; i < n; i++) {
        sum += *(arr+i);
    }
    return sum;
}

void functionAddress() {
    printf("Address is: %p", getSum);
}

int main() {
    functionAddress();
    return 0;
}
```

OUTPUT:

```
[samitdas:Downloads/ $ ./a.out
 Address is: 0x1024d3ed0
```

5. WAP to call main() using pointer to main() function.

CODE:

```
#include<stdio.h>
int flag = 0;
void callMain();

int main() {
    printf("Main function called, flag = %d\n", flag);
    if(flag == 1) {
        return 0;
    }
    flag = 1;
    callMain();
    return 0;
}

void callMain() {
    printf("callMain function is called\n");
    int (*main_fun)() = &main;
    (*main_fun)();
}
```

OUTPUT:

```
[samitdas:Downloads/ $ ./a.out
Main function called, flag = 0
callMain function is called
Main function called, flag = 1
```

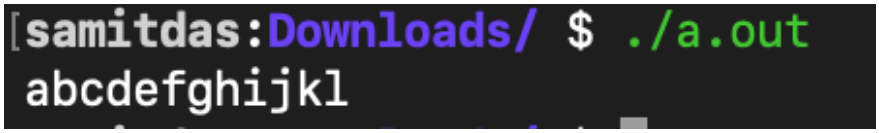
6. WAP to complete function *trim_blanks,* whose purpose is to take a single string input parameter (*to_trim*) and return a copy of the string with leading and trailing blanks removed.
Use *strncpy* in *trim_blanks*.

CODE:

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

int countNonSpace(char *str, int n) {
    int count = 0;
    for(int i = 0; i < n; i++) {
        if(str[i] ≠ ' ') {
            count++;
        }
    }
    return count;
}

char* firstNonSpace(char *str, int n) {
    for(int i = 0; i < n; i++) {
        if(str[i] ≠ ' ') {
            return str + i;
        }
    }
    return str + n;
}

void trim_blanks(char* str, int n) {
    int non_space = countNonSpace(str, n);
    char *src = firstNonSpace(str, n);
    char dest[200];
```

```
    memset(dest, '\0', sizeof(dest));
    strncpy(dest, src, non_space);
    printf("%s\n", dest);
}

int main() {
    char str[100] = "    abcdefghijkl    \0";
    trim_blanks(str, 20);
    return 0;
}
```

OUTPUT:



7. WAP to complete a function *string_greater* that could be used to find out-of-order elements when alphabetizing a list of strings in a situation in which the case of the letters should be ignored. Write a function *string_toupper* that converts each of its arguments to all capital letters before comparing them.

CODE:

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

void convert(char *s, char *c) {
    int j = 0;
    for(int i = 0; i < 25; i++) {
        if(s[i] == '\0') {
            break;
        }
        if(s[i] ≥ 'a') {
            c[j] = s[i] - 'a' + 'A';
        } else {
            c[j] = s[i];
        }
        j++;
    }
}

int func(char * s1, char * s2) {
    char c1[25];
    char c2[25];
    convert(s1, c1);
    convert(s2, c2);
    return strcmp(c1, c2);
}
```

```
int main() {
    char arr[50][25], temp[25];
    int n;

    printf("Input number of strings: ");
    scanf("%d", & n);

    printf("Input string %d :\n", n);
    for(int i = 0; i ⩽ n; i++) {
        fgets(arr[i], sizeof arr, stdin);
    }

    for(int i = 0; i ⩽ n; i++) {
        for(int j = 1; j ⩽ n; j++) {
            if(func(arr[j], arr[j – 1]) < 0) {
                char temp[25];
                strcpy(temp, arr[j]);
                strcpy(arr[j], arr[j – 1]);
                strcpy(arr[j – 1], temp);
            }
        }
    }

    printf("\nThe strings appears after sorting :\n");
    for (int i = 1; i ⩽ n; i++) {
        printf("%s", arr[i]);
    }
}
```

OUTPUT:

```
samitdas:Downloads/ $ ./a.out
Input number of strings: 5
Input string 5 :
BeSU
aPpLe
cAt
doG
Apz

The strings appears after sorting :
aPpLe
Apz
BeSU
cAt
doG
```