PRF 192
Workshop 4
Name: Hoang Thuy Nguyen
ID: DE191056

1. **Press “a” to generate an array containing n integers where each element is the sum of previous two integers.**
   Example: input 7, the output is 1 1 2 3 5 8 13
   a. Function:

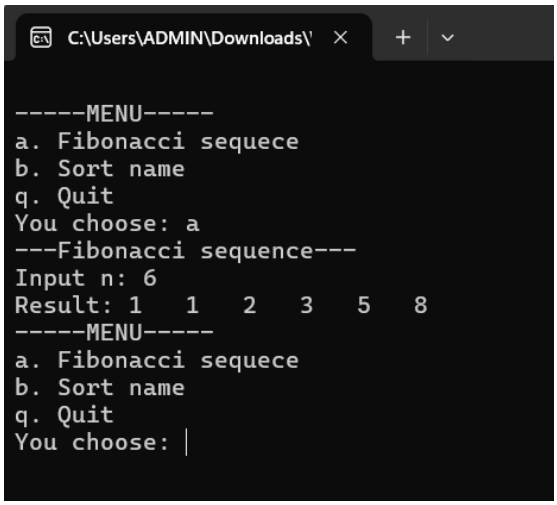
```c
//function a(Fibonacci)
int *fibo(int a) {
    int *p;
    *(p+1) = 1;
    *(p+2) = 1;
    for (int i=3; i<=a; i++) {
        *(p+i) = *(p+i-1) + *(p+i-2);
    }
    for (int i=0; i<a; i++) {
        *(p+i) = *(p+i+1);
    }
    return p;
}

```

   b. Call the function in main function:

```c
        case 'a': {
            int n;
            int *ptr;
            printf("---Fibonacci sequence---");
            printf("\nInput n: ");
            scanf("%d", &n);
            ptr = (int*)calloc(n,sizeof(int));
            ptr = fibo(n);
            printf("Result: ");
            for (int i=0; i<n; i++) {
                printf("%d   ", *(ptr+i));
            }
            break;
        }
```

c. Test:

- Case 1:



Walkthrough:

Line 85: enter 6→n=6

Line 87: pass n=6 into "fibo" function

Line 8: a = 6

Because after executing this program , we will receive an array of numbers with the principles of the problem math so I use pointer instead of array because I can use dynamic memory allocation for it(Line 86)

The principle of the problem math is that the next number is equal to the sum of the two previous numbers

Line 10: assign 1 to *(p+1) ← the second element of array

Line 11: assign 1 to *(p+2) ← the third element of array

Line 12: build for loop , let i run from 3 to a(a=6)

When i = 3 → *(p+3) = *(p+3-1) + *(p+3-2) = 1 + 1 = 2

I = 4 → *(p+4) = *(p+4-1) + *(p+4-2) = 1 + 2 = 3

…

Line 15: this loop will delete the first element, it mean *(p) from array

Line 18: function return pointer p

Line 87: assign value of pointer p of "fibo" function into pointer ptr of main function

Line 89 → line 91: print the result on screen

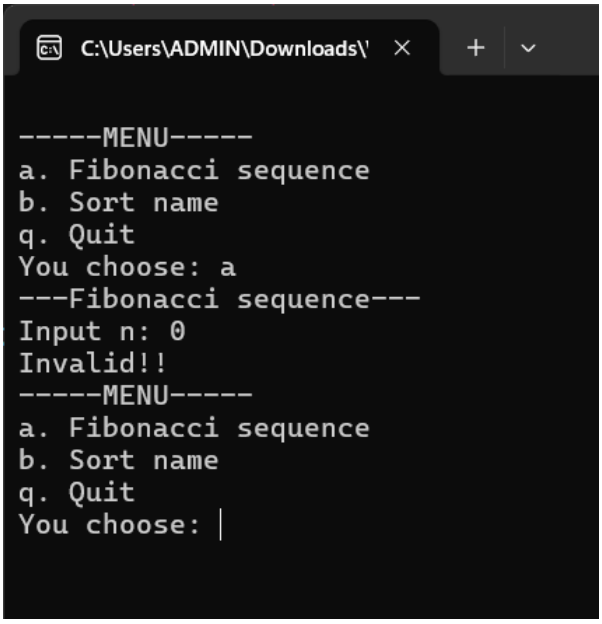
- Case 2:

The function:

```c
//function a(Fibonacci)
int *fibo(int a) {
    int *p;
    if (a>0) {
        *(p+1) = 1;
        *(p+2) = 1;
        for (int i=3; i<=a; i++) {
            *(p+i) = *(p+i-1) + *(p+i-2);
        }
        for (int i=0; i<a; i++) {
            *(p+i) = *(p+i+1);
        }
    } else {
        *(p) = -1;
    }
    return p;
}

```

Call function in the main:

```c
            case 'a': {
                int n;
                int *ptr;
                printf("---Fibonacci sequence---");
                printf("\nInput n: ");
                scanf("%d", &n);
                ptr = (int*)calloc(n,sizeof(int));
                ptr = fibo(n);
                if (*(ptr) == -1) {
                    printf("Invalid!!");
                } else {
                    printf("Result: ");
                    for (int i=0; i<n; i++) {
                        printf("%d   ", *(ptr+i));
                    }
                }
                break;
            }
```



Walkthrough:
Line 89: enter 0 → n=0
Line 90: dynamic memory allocation for pointer ptr
Line 91: pass n into fibo function
Line 8: n=0 → a=0
Because a=0 → Line 19: assign -1 to *(p)
Line 22: the function return pointer p
Line 91: assign result of fibo function to pointer ptr of main
Because (p) = -1 →*(ptr) = -1 →Line 93: print out on screen “Invalid!!”

2. **Press “b” to sort a given array of names to increasing alphabet. Test your code in main function.**
   Example: nameArray = {“Nguyen A”, “Le C”, “Doan E”, “Tran B”, “Huynh D”}, then array after sorted ={“Nguyen A”, “Tran B”, “Le C”, “Huynh D”, “Doan E”}

a. Function:

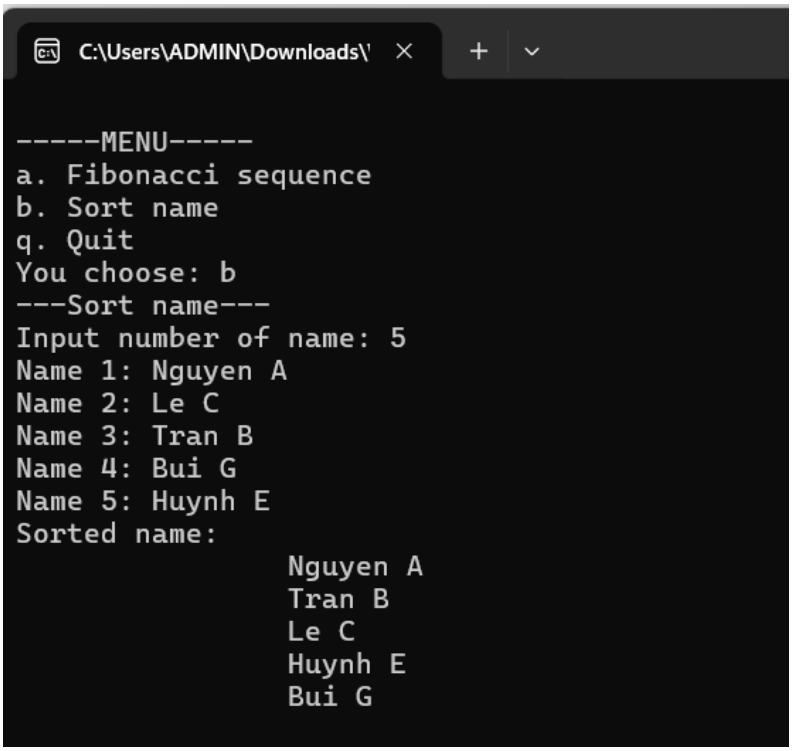
```
41  //sort name according to increasing alphabet
42  void sort_name(int a, char name[60][60], char temp_sen[60][60]) {
43      char *temp;
44      char te_sen[60][60];
45      temp = (char*)calloc(a,sizeof(char));
46      //search the first letter element and save it to pointer "temp"
47      for (int i=0; i<a; i++) {
48          for (int j=strlen(name[i]); j>=0; j--) {
49              if (name[i][j-1] == ' ') {
50                  *(temp+i) = name[i][j];
51                  strcpy(temp_sen[i],name[i]);
52                  break;
53              }
54          }
55      }
56      //swap
57      for (int i=a-1; i>0; i--) {
58          for (int j=0; j<=i-1; j++) {
59              char te;
60              if (*(temp+j) > *(temp+j+1)) {
61                  te = *(temp+j);
62                  *(temp+j) = *(temp+j+1);
63                  *(temp+j+1) = te;
64                  //swap name
65                  strcpy(te_sen[j],temp_sen[j]);
66                  strcpy(temp_sen[j],temp_sen[j+1]);
67                  strcpy(temp_sen[j+1],te_sen[j]);
68              }
69          }
70      }
71  }
72
```

b. Call the function in main function:

```
102             case 'b': {
103                 int n, cek;
104                 char NAME[60][60];
105                 char RS[60][60];
106                 printf("---Sort name---\n");
107                 printf("Input number of name: ");
108                 scanf("%d",&n);
109                 getchar();
110                 for (int i=0; i<n; i++) {
111                     printf("Name %d: ",i+1);
112                     gets(NAME[i]);
113                 }
114                 cek = check_name(n,NAME,RS);
115                 if (cek == 1) {
116                     sort_name(n,NAME,RS);
117                     printf("Sorted name: \n");
118                     for (int i=0; i<n; i++) {
119                         printf("\t\t%s\n",RS[i]);
120                     }
121                 } else {
122                     printf("Invalid name!!");
123                 }
124                 break;
125             }
```

c. Test:

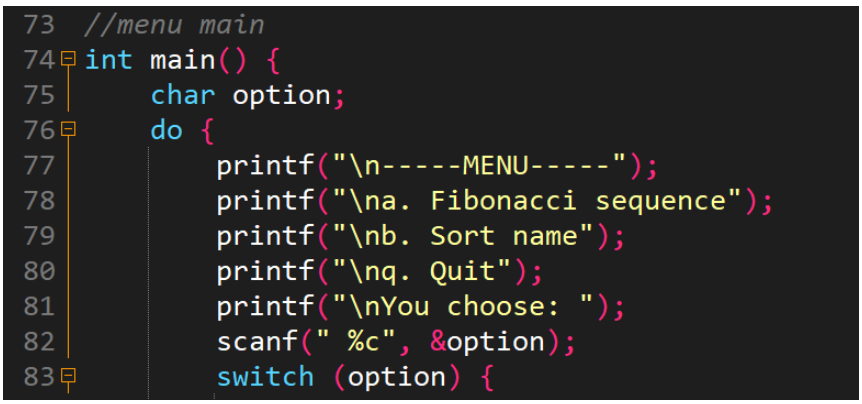
- Case 1:



Walkthrough:
Line 108: enter 5 → n=5
Line 109: Getchar function to consume "enter" character
At here, to store a name (including first name, middle name and last name) we use 2D array, so at line 104 and line 105 we declare NAME and RS with 2D array

With NAME array, we use to enter name from user
With RS array, is used to store sorted names

Line 110: for loop is used to input string name
Line 116: n = 5 , NAME and RS 2D array to sort_name function
Line 42: a=5, name 2D array store NAME array, temp_sen 2D array store RS array
Line 47 → line 55: use j and i loop to store the first letter of each name (only name. Example: Nguyen Lan Anh → store "A", Phan Thiet → store "T"). To store the first letters, I used pointer "temp" (with role is like normal array). Simultaneously, store entire name(including last, middle, first name) into "temp_sen", use strcpy to perform it
Line 57→line 71: continue to use to i and j loop to sort the first letters in pointer "temp" in ascending order, and at that time swap name elements in "temp_sen".
Line 116: sorted name in "temp_sen" array will be stored in RS array
Line 118 → line 119: print sorted names on screen

3.


```
73  //menu main
74  int main() {
75      char option;
76      do {
77          printf("\n-----MENU-----");
78          printf("\na. Fibonacci sequence");
79          printf("\nb. Sort name");
80          printf("\nq. Quit");
81          printf("\nYou choose: ");
82          scanf(" %c", &option);
83          switch (option) {
```

Line 83: enter any character to choose program which you want to computer execute
If you want to exit program you must enter ‘q’ →exit. Unless the program will loop

```
125                 }
126                 case 'q': {
127                     printf("Quit!");
128                     break;
129                 }
130                 default: {
131                     printf("Please choose again");
132                     break;
133                 }
134             }
135     } while (option != 'q');
136     return 0;
137 }
```

Case you want to exit:

```
-----MENU-----
a. Fibonacci sequence
b. Sort name
q. Quit
You choose: q
Quit!
--------------------------------
Process exited after 5.32 seconds with return value 0
Press any key to continue . . .
```

Case you enter wrong character: