CS266

ASSIGNMENT 10

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202051213

SECTION:

2

Code

```
/*
Archit Agrawal
202051213
*/

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

int fibonacci[10]; //stores the terms of fibonacci sequence

void main(){

    //initialising the fibonacci array
    fibonacci[0] = 8; //first two terms are to be considered 8 and 13
    fibonacci[1] = 13;
    for(int i = 2; i < 10; i++){
        fibonacci[i] = fibonacci[i - 1] + fibonacci[i - 2];
    }

    int index;
    printf("Enter a number between 1 and 10 to find the total memory size :
");
    scanf("%d", &index);
    //maximum memory limit
    int memory = fibonacci[index - 1];
    printf("\nTotal Memory Size is : %d units\n", memory);

    int allocated[fibonacci[index - 1]];  ///stores the space that is
allocated and that is free
    //allocated[i] = positive integer means space is allocated to a process
with process id = positive integer
    //allocated[i] = -1 means space is free
    //allocated[i] = 0 means space has hole


    int x = 0;
    while(x != -1){
        printf("To allocate space to a process, enter 1.\n");
        printf("To deallocate space of a process, enter 2.\n");
        printf("To exit, enter -1\n");

        scanf("%d", &x);

        if(x == 1){
```

```
            printf("Enter process ID : \n");
            int id;
            scanf("%d", &id);
            // to ensure space is between 8 and the maximum memory limit
            int space = (rand() % (memory - 8)) + 8;
            printf("Space Required by process %d is %d : \n", id, space);
            allocate(allocated, space, id, index - 1, memory);
        } else if(x == 2){
            int id;
            printf("Enter process ID : \n");
            scanf("%d", &id);
            deallocate(allocated, id, memory);
        }
    }
}

void allocate(int allocated[], int space, int id, int index, int sum){
    if(index < 2 && space < fibonacci[index]){
        sum = sum - fibonacci[index];
        int canBeAllocated = 0;
        for(int i = sum; i < sum + space; i++){
            if(allocated[i] > 0){
                canBeAllocated = -1;
                break;
            }
        }
        if(canBeAllocated != -1){
            printf("Space Allocated to process %d from %d to %d\n", id, sum,
sum + space);
            printf("Holes are generated from %d to %d\n", sum + space, sum +
fibonacci[index]);
            for(int i = sum; i < sum + space; i++){
                allocated[i] = id;
            }
            for(int i = sum + space; i < sum + fibonacci[index]; i++){
                allocated[i] = -1; //generated a hole
            }
        } else {
            printf("Space can't be allocated.\n");
        }
    } else if(space < fibonacci[index - 2]){
        /*
        fibonacci[index - 2] will definitely be lesser than fibonacci[index -
1]
        hence, if space is less than fibonacci[index - 2],
        then we can recurse down to the smaller section
        */
```

```
        sum = sum - fibonacci[index - 2];
        allocate(allocated, space, id, index - 2, sum);
    } else if(space > fibonacci[index - 2] &&  space < fibonacci[index - 1]){
        /*
        space required is greater than the smaller section
        only option is to recurse down the larger section.
        */
        sum = sum - fibonacci[index - 1];
        allocate(allocated, space, id, index - 1, sum);
    } else if(index >= 2 && space > fibonacci[index - 1] && space >
fibonacci[index - 2]){
        sum = sum - fibonacci[index];

        int canBeAllocated = 0;
        for(int i = sum; i < sum + space; i++){
            if(allocated[i] > 0){
                canBeAllocated = -1;
                break;
            }
        }
        if(canBeAllocated != -1){
            printf("Space Allocated to process %d from %d to %d\n", id, sum,
sum + space);
            printf("Holes are generated from %d to %d\n", sum + space, sum +
fibonacci[index]);
            for(int i = sum; i < sum + space; i++){
                allocated[i] = id;
            }
            for(int i = sum + space; i < sum + fibonacci[index]; i++){
                allocated[i] = -1; //generated a hole
            }
        } else {
            printf("Space can't be allocated.\n");
        }
    }
}

void deallocate(int allocated[], int id, int memory){
    for(int i = 0; i < memory; i++){
        if(allocated[i] == id){
            allocated[i] = -1;
        }
    }
    printf("Space Deallocated\n");
}
```

OUTPUT

```
Enter a number between 1 and 10 to find the total memory size : 10

Total Memory Size is : 610 units
To allocate space to a process, enter 1.
To deallocate space of a process, enter 2.
To exit, enter -1
1
Enter process ID :
1
Space Required by process 1 is 49 :
Space can't be allocated.
To allocate space to a process, enter 1.
To deallocate space of a process, enter 2.
To exit, enter -1
2
Enter process ID :
1
Space Deallocated
To allocate space to a process, enter 1.
To deallocate space of a process, enter 2.
To exit, enter -1
1
Enter process ID :
2
Space Required by process 2 is 415 :
Space can't be allocated.
To allocate space to a process, enter 1.
To deallocate space of a process, enter 2.
To exit, enter -1
-1
PS C:\Users\Archit\Desktop\cprog>
```