Assignment - 1: Memory Management

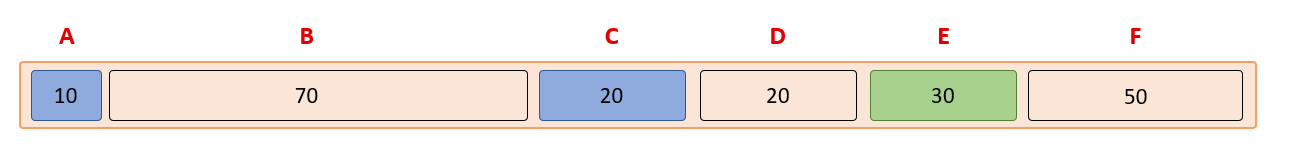
Due Date: 11.59 PM, February 13th (Saturday Night)

Total Marks: 20 marks

Memory management policies:

1. First Fit: Allocates the first sufficient free memory block from the start of main memory.
2. Next Fit: Similar to the first fit policy, but allocates the first sufficient free memory block starting from the last allocation point. Move back to the beginning of the main memory, once you reach the end.
3. Best Fit: Allocates the smallest sufficient memory block among the available free memory blocks. If there are multiple blocks with the same smallest size, choose the first one.

Consider the following example. Blue blocks (A and C) represent allocated memory blocks. The green block (E) is the last allocated memory block.



When the user requests for a memory block of size 10 bytes,

* First Fit allocates 10 bytes from block B.
* Next Fit allocates 10 bytes from block F.
* Best Fit allocates 10 bytes from block D.

*Note: Space occupied by the book-keeping structure is not represented in the above diagram.*

Functions to be implemented as part of the assignment:

1. void allocate (int n)

* Allocates memory of n bytes to the global character array.
* The memory allocated by mymalloc () function + memory needed to store the book-keeping structure must be allocated within those n bytes itself.

1. void mymalloc (int size)

* Allocates ‘size’ bytes from the available memory and returns a pointer pointing to this memory block.
* Allocate memory based on the memory allocation policy assigned to you, i.e, FirstFit, NextFit, or BestFit.
* If no free memory block is available, return NULL.

1. void myfree (void\* b)

* Free the block of memory pointed by pointer ‘b’.
* Should merge free blocks adjacent to the newly freed block.
* **Note: Only valid pointers will be passed as parameters to the function.**

1. void print\_book\_size ( )

* Prints the size of the book-keeping structure.
* Use the following print statement in this function

*printf(“%ld\n”,size\_of\_structure);*

* **Do not include any other code in this function. Do not change the above print statement.**

1. void display\_mem\_map ( )

* Display the view of memory at any instance it is called.
* Structure of output:

Start\_address<\t>block\_in\_bytes<\t>encoded\_status<\n>

* Encoded\_status : 0 - book, 1 - allocated, 2 - free

**Your output should be similar to the screenshots below. Do not include any other unnecessary print statements. If the format is not followed, you will receive a zero for this assignment.**

// Assume n = 100, size of book-keeping structure = 16

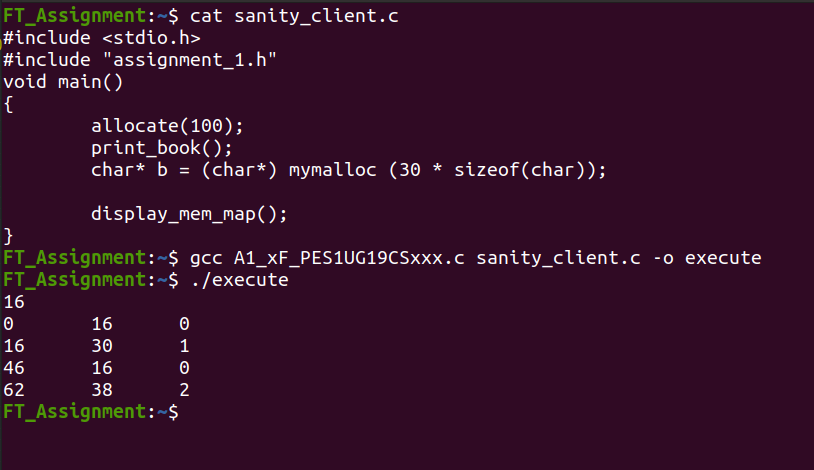
**// Print ‘\t’ between the fields**

// Output of display\_mem\_map ( ) if called in the start (after a call to allocate(100); )

0 16 0

16 84 2

Sample outputs with sanity\_client.c



**Note:** In case there is no space to store the book-keeping structure in the remaining free space after memory allocation, the remaining free space must be attached to the newly allocated block. The same must be reflected in the output of display\_mem\_map ( ). The following output explains the same.



In the above example, 8 bytes of memory is left after allocating memory to pointer c. Since 16 bytes are required to store a book-keeping structure, the 8 bytes are allocated to pointer c. Hence c is assigned 38 bytes (instead of 30 bytes).

Deliverables:

Each student will implement one allocation policy. The list of allocation policies are:

1. First Fit (FF)
2. Next Fit (NF)
3. Best Fit (BF)

((last three digits of SRN) % 3) + 1 will be the allocation policy you will be implementing.

Each student is required to submit 2 files ( .c file and .txt file)

1. A1\_<allocation\_policy>\_<SRN>.c

* This should contain the implementation for the above functions.
* Please ensure that if you are using any helper functions in your implementation file, they are named with the **static** keyword so they don't clash with any functions that we will use to test your code.
* Please comment your code as well (you will be graded based on this as well).
* File naming format: A1\_FF\_PES1UG19CS000.c, A1\_NF\_PES1UG19CS000.c, A1\_BF\_PES1UG19CS000.c for first fit, next fit and best fit respectively.

1. A1\_README\_<SRN>.txt: This should contain the following things:

* Allocation policy used
* Basic logic used for implementation
* Explanation of the book-keeping structure
* Any additional notes.
* A header file (assignment\_1.h) has been provided which contains the function definitions. **You are not allowed to change these function definitions.**
* *extern char\* p;* Pointer to the global char array.
* *extern int size;* Size of the global char array.
* A sanity\_client.c file has been provided which can be used to check if your implementation works. Ensure that your code generates the expected output for the sanity\_client.c before submitting. **We will be testing your implementation file for various other test cases as well.**
* Codes submitted with compilation errors will directly be rejected (zero marks).
* Comments where necessary, proper indentation, and meaningful variable names will also carry marks.
* In addition to the header file provided, please ensure that you only include header files in GCC’s standard collection.
* Do not deviate from the expected output format. Unnecessary print statements will result in a wrong answer.
* Do not use pragma pack to modify structure padding.
* Code written by NSK sir in class (if any) can be used for your assignment.

You can ask your doubts at [Doubt Clarifications](https://docs.google.com/spreadsheets/d/1pMnjK6HXHZYLRXddQPp3yp0hkXvw0saxsXYrKcmJNu0/edit?usp=sharing)

Submission link: (Will be updated)

**Any cases of plagiarism will be dealt with seriously. DO NOT COPY/SHARE CODE. Do not take code from your peers or seniors or from online sources.**

**If you believe you can get away after plagiarising, think again.**