**Some C Programming Language Tips:**

(1)

**#define TEST //**The Green colored section want be compiled except this line is been written

**Program(){**

……………………………

………………………………

…………………………..

#ifdef TEST

……………………………..

……………………………..

………………………………

#end if

……………………………..

……………………………..

**}**

(2)

Note: “The Green colored section will never be compiled

…………………………

…………………………………………

……………………………………………

#if 0

………………………………..

………………………………..

………………………………...

#end if

(3)

How to make a pointer to a function??

Void myfun (int x){

……………………………

……………………………

…………………………...}

Void (\*foo) (int);

Foo= &myfun;

Foo(2); // will act exactly the same as myfun(2)

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**Famous Question of SySDSoft:**

1. **How can processes communicate?**

**Answer:**

Process A

Process A

Shared Memory

Process B

Kernel

Process B

**Message Passing Shared Memory**

1. **What is the difference between Hard Real Time and Soft Real Time?**

A hard real-time system is a system whose operation is incorrect if results are not produced according to the timing specification.

**EX:** Safety-critical system

* + 1. Ex: Flight Control System
    2. Ex: Airbag crash detection system

A soft real-time system is a system whose operation is degraded if results are not produced according to the specified timing requirements.

**EX:** Non-safety-critical system

1. Keypad input
2. Message visualization
3. System status representation
4. **What is Meant by Function Reentrancy?**

It’s the characteristic that describes the ability of a function to be recalled by another source while the first call hasn’t been finished yet, it might occur during the normal context switching between tasks that deal with the same function.

A function is called reentrant if it has no static variables, and it doesn’t call any other non-reentrant function.

1. **What is the difference between Real Time Systems and Normal Systems?**

Real Time Systems needs for deterministic timing behavior. We ento seed el 3arfeen ya3ni we 7ate3rafo tegawboh be kafa2a isa.

1. **What is meant by Cache Memory, Paging, and Pipelining?**

**Paging 🡪** Allows OS to retrieve data from secondary storage in the same size blocks called pages. Allows OS to use disk for data that doesn’t fit into RAM.

**Cache Memory 🡪**  reduces Predictability, It’s not used in Hard Real Time Systems “walahy ma3raf leih we yareit elli ye3raf ye2ool”.

**Pipelining 🡪** is advanced micro processing technique capable of handling multiple stages of different computer instructions at one time; this allows the performance and overall throughput to be enhanced.

1. **What is meant by Thread, Process, Preemption, and Kernel?**

**Thread 🡪** It does exist as a subset of process preemption while multithreads occurs by time division multiplexing of the process switches between different threads.

**Kernel 🡪**  It’s the core of the OS which is responsible for communication between the hardware and the software overall all software layers of the system, in addition to performing the inter process communication.

**Process and Preemption 🡪** Process is the single task of the software that can be handled separately that represents the background behavious of the software “zai el task tamam zai ma2al 3amr 3adel lamma negi netakllem 3annaha in general we el preemption ento seed el 3arfeen”

1. **What is the difference between Structures and unions??**

Although when both are compiled, no memory reservation takes place unless a variable of this time is been declared, Union’s fields affect each others while structure doesn’t.

i.e. Unions reserve a memory space of the largest variable type that is contained by it, this memory space is been used by all variables “once a time.

Example:

Union mytype{

Int A;

Char B;

Char Name[50];

}

**B**

This 50 bytes can be used for the Char B

**A**

**OR for the Int A**

**OR** it can be used for the 50 bytes of the character Array

1. **What is the difference between Stack and Array?**

**Stack :** The is no direct access to the stack elements “ lazem a3mel pop le element wara el tani le7ad mawsal lelli ana 3awzoh”; It’s size is not predetermined “ya3ni kol ma3ooz a3mel push a3mel 3adi we mosh mo7addad be length mo3ayan (tab3an mosh asdi 3ala el stack allocation elli 2al 3aleih Eng. Tamer)”

**Array :** The is direct access to the array elements “ sahl awy 2a2ol arr[5] aw arr[8]”; It’s size is predetermined “ya3ni hena wana ba3mel decalaration lel array lazem 2akoon me7aded el length leha 7atta law mastakhdemtesh kol el amaken di”

1. **What is the difference between the array and the linked list?**

|  |  |
| --- | --- |
| **ARRAY** | **LINKED LIST** |
| The size of the array is static “Predetermined” (Fixed Length) | It has an unlimited list size “Dynamic” (variable length) |
| Very hard to delete or insert an element as a shift process for all the successive elements is required | Requires only two steps to insert or delete an elements. “Insert: akhalli el pointer beta3 elli abl el inserted element yeshawer 3aleih, we ba3deen akhalli el pointer beta3 el inserted elemenet yeshawer 3alli ba3doh, Delete: akahlli el pointer beta3 elli abl el deleted element yeshawer 3alli ba3doh, we ba3deen amsah el element dah “free(ptr)”” |
| It might cause a waste of memory spaces if all the length is not filled with elements as their spaces are reserved anyway | Linked list that contains integers use a double space of memory of an array of the same length, “la2en kol element ba2a value”int” we pinter” |

1. **What is meant by macros?**

Macro is a type of preprocessing that is similar to #define, on the other hand that it’s a function itself.

Example:

#define ADD(x,y) x+y; /\* so whenever we write ADD(5,6) it will exactly as we are calling a function that takes 2 arguments, summate them, then return the summation result\*/

//the main idea of macros is that, it’s a preprocessing and that we don’t have //to determine the type of neither x nor y.

1. **How can you sort an array entered by a user without any element repetition ? “ yagama3a dah el 7al beta3i, we 3aref walahi enni 7al mo3awak :D :D :D we en ento masha2 allah 7ate3rafo te3melolo optimization a7san men keda beketeer”**

#include <stdio.h>

#include <conio.h>

void bbl\_Sort(int \* ptr)

{

int flag=1;

int temp,i;

while(flag)

{

flag=0;

for(i=9;i>=1;i--)

if(ptr[i]<ptr[i-1])

{

temp=ptr[i];

ptr[i]=ptr[i-1];

ptr[i-1]=temp;

flag++;

}

}

}

Int no\_Rep (int \* ptr)

{

int ptr\_array\_index=0;

int temp[10],i;

for(i=0;i<=9;i++)

temp[i]=ptr[i];

ptr[ptr\_array\_index]=temp[0];

ptr\_array\_index++;

for(i=1;i<=9;i++)

if(temp[i]==temp[i-1])

continue;

else

{ptr[ptr\_array\_index]=temp[i]; ptr\_array\_index++;}

return (ptr\_array\_index-1);

}

int main (void)

{

int arr[10];

int i, No\_of\_elements\_without\_rep;

while(1)

{

printf("\nPlease enter 10 elements of your choice then press enter to get them sorted \n");

for(i=0;i<=9;i++)

{

scanf("%d",&arr[i]);

}

bbl\_Sort(arr);

No\_of\_elements\_without\_rep=no\_Rep(arr);

printf("\n The sorted array of your entry is:\n");

for(i=0;i<=No\_of\_elements\_without\_rep;i++)

printf("%d ",arr[i]);

printf("\nPlease press any key to proceed\n");

getch();

}

return 0;

}

1. **How can yoy manage accessing a certain element in a linked list?**

**“yagama3a dah el linked list kolaha we 7atla2o westaha ezai a3mel accessing for a certain element, Please find the attached .cpp file named Linked List”**

1. **What is the difference between stack and heap?**

**Stack:**

* Stored in computer RAM like the heap.
* Variables created on the stack will go out of scope and automatically deallocate.
* Much faster to allocate in comparison to variables on the heap.
* Implemented with an actual stack data structure.
* Stores local data, return addresses, used for parameter passing
* Can have a stack overflow when too much of the stack is used. (mostly from inifinite (or too much) recursion, very large allocations)
* Data created on the stack can be used without pointers.
* You would use the stack if you know exactly how much data you need to allocate before compile time and it is not too big.
* Usually has a maximum size already determined when your program starts

**Heap:**

* Stored in computer RAM like the stack.
* Variables on the heap must be destroyed manually and never fall out of scope. The data is freed with delete, delete[] or free
* Slower to allocate in comparison to variables on the stack.
* Used on demand to allocate a block of data for use by the program.
* Can have fragmentation when there are a lot of allocations and deallocations
* In C++ data created on the heap will be pointed to by pointers and allocated with new or malloc
* Can have allocation failures if too big of a buffer is requested to be allocated.
* You would use the heap if you don't know exactly how much data you will need at runtime or if you need to allocate a lot of data.
* Responsible for memory leaks

1. **What are the scheduling techniques??**
2. First-come First served
3. Shortest Job First
4. Priority
5. Round-Robin “we akeed el kalam wade7 leena isa”
6. **What is the difference between static and dynamic allocation?**

Dynamic memory allocation is at runtime. static memory allocation is before run time, but the values of variables may be changed at run time.   
Static memory allocation saves running time, but can't be possible in all cases.   
Dynamic memory allocation stores it's memory on heap, and the static memory allocation stores it's data in the "data segment" of the memory.

Static memory allocation is all the time since it’s a RAM or ROM variables while dynamic memory allocation is Heap or stack; i.e. has a life time.

1. **What is the difference between Pointers and dynamic allocation?**

**“Eng. Ahmed Loutfy’s Answer”**

Pointer is a data type itself that is used to refer to what it points to, or the address it contains; the pointer itself is a static allocation when it’s declared before compiling “in the code”

The dynamic allocation is used to reserve a piece of memory during the runtime and return only the first address “reference” of this piece as a pointer to void …so we need a pointer and a casting process to save this reference to be able to deal with this piece of memory.