SOFE 3950U: Tutorial 7 Activity
Group 1

Justin Kaipada 100590167
Anthea Ariyajeyam 100556294

Conceptual Questions

Question 1

Signals are one of the most simplified forms of IPC mechanism, where no data transfer occurs. Signals are asynchronous interrupt of the target process or thread which will have to handle the signal with a specific routine during non-atomic operations. Signals are started by the kernel and handled by the processes. The computational and memory footprint of signals are hence small.

Question 2

SIGINT - This is the signal sent when we want to interrupt a process, from the terminal you can use Ctrl-C, to send the signal to the current process.

SIGTSTP - This is the signal sent to stop a process for now in manner so it can be resumed later. We can use Ctrl-z to send a SIGTSTP to the current process. We can also use the kill utility to send this signal like this:

```
kill -SIGSTOP [pid]
```

SIGCONT - This is the signal sent to continue or restart a process that was stopped using SIGSTP . We can use the kill utility to send this signal like this:

```
kill -SIGCONT [pid]
```

Question 3

kill() - is a system call used to send any signal to a process or processes the current program have permission for. In C a signal can be sent using the following syntax. kill(pid,sig) , where pid is the process id and sig is the signal we are trying to sent.

waitpid() - is a system call used to wait for a child process to change its state. The state change can be a termination or being stopped or being resumed after stopping. When using waitpid(pid,status,options) , pid is the process id the child-process being waited for, status is the type of exit status or state changes we are waiting for and options specify optional actions for the waiting which can also be a combination of many options.

To terminate a child process and wait until it has ended we can do something like this in C:

```
1 //Create a child
int status;
pid_t child2;
child1 = fork();
// Do something with child
 ...
// Child is dead waitpid(-1, &status, 0); // Wait until child is dead here
```

Question 4

A link list is a data structure that contains a node that points to the next item (i.e. data) in the list and the data. However, the first item in the list will point to head which represents a dummy node and used to indicate the beginning of the list.The common operators that a link list must have include:

- Pop which must remove the item
- Push which must add an item to the end of the list

FIFO means the item that first enters a list will be the first to exit.

Question 5

A link list implemented in C would use a struct that contains a pointer to the next item in the list and item in the list.

E.g. stuct linkList

```
{
        Int item;
        struct linkList *next;
}
```

To implement the operations the user is required to use create a pointer called head as a global variable.

E.g. struct linkList *head;

To add values to the link list the data and the head must be assigned to a pointer. The data is assigned to the item in the list and head is assigned to the pointer (that points to the next item).

E.g. struct queue *temp = malloc(sizeof(struct linkList));

```
temp->item = data;
temp->next = head;
```

Finally, the pointer must be assigned to the head

E.g. head = temp

To remove value in the link list you must find the location of the item to be removed and make its previous item point to the next item of the item to be removed. Finally, pointer for the item to be removed is pointed to NULL.