1.
They are part of the environment in which a process runs.
Its variable is a dynamic-named value that can affect the way running processes will behave on a computer.
Environment variables hold values related to the current environment, like the Operating System or user sessions.

a)
```
#include <stdio.h>
#include <stdlib.h>

int main ()
{

   printf("USER : %s\n",getenv("USER"));
   printf("HOME : %s\n", getenv("HOME"));
   printf("PWD : %s\n",getenv("PWD"));
   printf("PATH : %s\n", getenv("PATH"));



   return(0);
}
```
OUTPUT:
```
USER : amrit
HOME : /home/amrit
PWD : /home/amrit/os/assign2
PATH : /home/amrit/bin:/home/amrit/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

b)
```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
   setenv("ROOT","root",1);
   printf("ROOT: %s\n", getenv("ROOT"));



   return(0);
}
```

OUTPUT:
```
        ROOT: root
```

2)
```
#include<stdio.h>
#include<pthread.h>
void* phello()
{
        printf("HELLO WORLD");

}


int main()
{
        pthread_t t1;
        pthread_create(&t1,NULL,phello,NULL);
        pthread_join(t1,NULL);
}
```
OUTPUT:
```
HELLO WORLD
```

b)
```
#include<stdio.h>
```

```
#include<stdlib.h>
#include<pthread.h>
int a,b,wait=0;
pthread_mutex_t mutex;
pthread_cond_t cv;

void * printa()
{
                //Critical Section
        pthread_mutex_lock(&mutex);
        if (b!=0)
        {       wait++;
                pthread_cond_wait(&cv,&mutex);
        }
        printf("a: %d\n",a);
        pthread_mutex_unlock(&mutex);

}
int main()
{
        pthread_mutex_init(&mutex, NULL);
        pthread_cond_init(&cv, NULL);
        pthread_t t1;
        pthread_create(&t1,NULL,printa,NULL);
                //Critical section
        pthread_mutex_lock(&mutex);
        printf("ENTER THE VALUE: ");
        scanf("%d",&a);
        b=0;
        if (wait!=0)
                pthread_cond_signal(&cv);
        pthread_mutex_unlock(&mutex);
        pthread_join(t1,NULL);


}
```

```
OUTPUT:
ENTER THE VALUE: 5
a: 5
```

3)

```
OUTPUT:
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000x
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

This program would exicute either the print_os or the thread function print_xs, Sice the code uses mutual exclsion principle they wont work alternatively in between.

This code will print 1000 ’o’ followed by that much of ’x’ mostly, wiseversa can also rarel
y happen, if the new thread was in the first exicution.

b)

```
#include <stdio.h>
#include <pthread.h>
static void print_os();
pthread_mutex_t mutex;
int wait=0;
pthread_cond_t cond;
void* print_xs(void* unused) {
        int j;
        pthread_mutex_lock(&mutex);
        for(j = 0; j < 1000; j++)
        fputc(’x’, stderr);
        wait++;
        pthread_cond_signal(&cond);
        pthread_mutex_unlock(&mutex);
        return NULL;
}
static void print_os() {
        int i;
        for(i = 0; i < 1000; i++)
        fputc(’o’, stderr);
}
int main() {
        pthread_mutex_init(&mutex, NULL);
        pthread_cond_init(&cond, NULL);

        pthread_t new;
        pthread_create(&new, NULL, &print_xs, NULL);
        pthread_mutex_lock(&mutex);
        if (wait<=0)
                pthread_cond_wait(&cond,&mutex);
        print_os();
        pthread_mutex_unlock(&mutex);
        pthread_join(new, NULL);
        return 0;
}
```

OUTPUT:

4)

```
#include<stdio.h>
```

```
#include<stdlib.h>
#include<pthread.h>
int wait=0;
pthread_mutex_t mutex;
pthread_cond_t cv;
signed long long int buf;
int wa=-1,wb=-1;

void * printa()
{//Critical Section
        signed long long int a,b;
        pthread_mutex_lock(&mutex);
        if (wait==0)
        {
                wait=2;
                pthread_cond_wait(&cv,&mutex);

        }
        printf("a recieved: %lli \n",buf);
        if(wa==0)
        {
                a=buf;
                buf=0;
                pthread_cond_signal(&cv);
                wb=0;}
        if(wb==0)
        {pthread_cond_wait(&cv,&mutex);
                printf("b recieved: %lli \n",buf);
                b=buf;
                buf=0;
        }
        buf=a*b;
        printf("buf:%lli\n",buf );
        printf("a:%lli,b:%lli\n",a,b);
        pthread_cond_signal(&cv);
        pthread_mutex_unlock(&mutex);

}
int main()
{ signed  long long int k[2];
        pthread_mutex_init(&mutex, NULL);
        pthread_cond_init(&cv, NULL);
        pthread_t t1;
        pthread_create(&t1,NULL,printa,NULL);

                //Critical section
        pthread_mutex_lock(&mutex);
        printf("ENTER THE Ist VALUE: ");
        scanf("%lli",&k[0]);
        buf=k[0];
        wa++;
        wait++;
        if (wait==2)
        {
                pthread_cond_signal(&cv);
        }
        if (wb!=0)
        {
                pthread_cond_wait(&cv,&mutex);
        }
        printf("ENTER THE  II nd VALUE: ");
        scanf("%lli",&k[1]);
        buf=k[1];

        wb++;
        if(wb==1)
        {
                pthread_cond_signal(&cv);
```

```
        }
        if (buf==k[1])
                pthread_cond_wait(&cv,&mutex);

        pthread_mutex_unlock(&mutex);
        printf("mul: %lli ", buf);
        pthread_join(t1,NULL);

}
```

```
OUTPUT:
ENTER THE Ist VALUE: 2
a recieved: 2
ENTER THE  II nd VALUE: 3
b recieved: 3
buf:6
a:2,b:3
mul: 6

amrit@mypc:~/os$ ./a.out
ENTER THE Ist VALUE: 55556
a recieved: 55556
ENTER THE  II nd VALUE: 2235
b recieved: 2235
buf:124167660
a:55556,b:2235
mul: 124167660
```