Answer

Program Screen shot:

```c
// Declare the required variable
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

// Create a function of the
// Collatz conjecture concern
int Collatz(int num)
{
// declare the variable
int count = 0;

// check the required error condition
if(num <= 0)
{
printf("<Starting Value> should be positive integer");
return 0;
}
/*if(num == )
{
printf("<Starting Value>");
return 0;
}
*/

// While loop repeats time 1000
//to execute the program
while(count != 10000)
{

// when input number is equal to 1
if(num == 1)
{
// display the number
printf("%i ",num);

// return the zero value
return 0;
}

// when number is even
else if(num % 2 == 0)
{

// display the even number
printf("%i ",num);

// half the number to even
num /= 2;
}
// when number is odd
else
{

// display the number
```

```
printf("%i ",num);

// number verified it is odd condition
num = num * 3 + 1;
}

// increment the counter value
count++;
}
return 0;
}

// create the main function
int main(int argc, char *argv[])
{

// declare the required variable
int num, stat_loc = 0;
pid_t pid;
num = atoi(argv[1]);

// declare the fork function
pid = fork();

// create the child function
// process id
if(pid == -1)
{

// Display when child process did not
// created
printf("Child Process didn't get created.");
}
// when child process id is 0
// then child process created
else if(pid == 0)
{

printf(": ");
Collatz(num);
}

// when process id is greater than 0
else if(pid > 0)
{
//printf("<starting value>");

// wait until the chil process finished
wait(&stat_loc);

printf("\n");
}
}
```

Sample output:

```
vikasbro@ubuntu:~/Desktop$ gcc vik.c
vikasbro@ubuntu:~/Desktop$ ./a.out 3
: 3 10 5 16 8 4 2 1
vikasbro@ubuntu:~/Desktop$ ./a.out -3
: <Starting Value> should be positive integer
```

Program code to copy:

```
// Declare the required variable
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

// Create a function of the
// Collatz conjecture concern
int Collatz(int num)
{
// declare the variable
int count = 0;

// check the required error condition
if(num <= 0)
{
printf("<Starting Value> should be positive integer");
return 0;
}
/*if(num == )
{
printf("<Starting Value>");
return 0;
}
*/
```

```
// While loop repeats time 1000
//to execute the program
while(count != 10000)
{

// when input number is equal to 1
if(num == 1)
{
// display the number
printf("%i ",num);

// return the zero value
return 0;
}

// when number is even
else if(num % 2 == 0)
{

// display the even number
printf("%i ",num);

// half the number to even
num /= 2;
}
```

```
// when number is odd
else
{

// display the number




printf("%i ",num);

// number verified it is odd condition
num = num * 3 + 1;
}

// increment the counter value
count++;
}
return 0;
}

// create the main function
int main(int argc, char *argv[])
{

// declare the required variable
```

```
int num, stat_loc = 0;

pid_t pid;

num = atoi(argv[1]);


// declare the fork function

pid = fork();


// create the child function

// process id

if(pid == -1)

{


// Display when child process did not

// created

printf("Child Process didn't get created.");

}

// when child process id is 0

// then child process created

else if(pid == 0)

{


printf(": ");

Collatz(num);

}


// when process id is greater than 0
```

```
else if(pid > 0)

{

//printf("<starting value>");


// wait until the chil process finished

wait(&stat_loc);


printf("\n");

}

}
```

Likes: 3 **Dislikes: 0**
