11. Write a C program to create 4 child processes. In the first child process, print the odd numbers. In the second child process print the even numbers. In the third child process print the multiple of 3. In the fourth child process print the multiples of 5. Print the process id for each of the processes.

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <sys/wait.h>

void print\_odd\_numbers()

{

pid\_t pid = getpid();

printf("Process %d printing odd numbers: ", pid);

for (int i = 1; i <= 10; i += 2)

{

printf("%d ", i);

}

printf("\n");

}

void print\_even\_numbers()

{

pid\_t pid = getpid();

printf("Process %d printing even numbers: ", pid);

for (int i = 2; i <= 10; i += 2)

{

printf("%d ", i);

}

printf("\n");

}

void print\_multiples\_of\_three()

{

pid\_t pid = getpid();

printf("Process %d printing multiples of three: ", pid);

for (int i = 3; i <= 30; i += 3)

{

printf("%d ", i);

}

printf("\n");

}

void print\_multiples\_of\_five()

{

pid\_t pid = getpid();

printf("Process %d printing multiples of five: ", pid);

for (int i = 5; i <= 50; i += 5)

{

printf("%d ", i);

}

printf("\n");

}

int main()

{

pid\_t child\_pid[4];

for (int i = 0; i < 4; i++)

{

child\_pid[i] = fork();

if (child\_pid[i] == 0) // child process

{

switch (i)

{

case 0:

print\_odd\_numbers();

break;

case 1:

print\_even\_numbers();

break;

case 2:

print\_multiples\_of\_three();

break;

case 3:

print\_multiples\_of\_five();

break;

default:

break;

}

exit(0);

}

else if (child\_pid[i] < 0)

{

printf("Fork error.\n");

exit(1);

}

}

// wait for all child processes to finish

for (int i = 0; i < 4; i++)

{

waitpid(child\_pid[i], NULL, 0);

}

return 0;

}

