1. **Write a program to create one child. Execute some code in the child such that the parent exits before the child. Showing that the child still remains under execution also mentions the new parent process ID of the child.**

```
#include<sys/types.h>
#include<unistd.h>
#include<stdio.h>

void child();
void parent();

int main()
{
pid_t pid;
pid=fork();
if(pid==0)
child();
else if(pid>0)
```

```
parent();
else
printf("\nfork is failed!!");

return 0;

}

void child()
{
printf("\nchild's PID= %d", getpid());

sleep(20);
printf("\nchild's parent's PID= %d",
getppid());

printf("\nChild is Exiting...\n");
}

void parent()
```

```
{
printf("\nparent's PID= %d",
getpid());
sleep(2);
printf("\nparent is Exiting...\n");
}
```

2. Write a program to create one child such that the child exists before execution of the parent completes. As long as the parent remains under execution, show the status of the child (terminated or exists). If the child still appears in the process list (at an instance when child execution is finished and parent is still under execution) then change your program such that, child is not found in the process list, as soon as its execution gets completed.

```
#include<sys/types.h>
#include<unistd.h>
#include<stdio.h>

void child();
void parent();
```

```
int main()
{
pid_t pid;
pid=fork();
if(pid==0)
child();
else if(pid>0)
parent();
else
printf("\nfork is failed!!");

return 0;

}

void child()
{
printf("\nchild's PID= %d", getpid());
```

```
sleep(2);
printf("\nchild's parent's PID= %d",
getppid());

printf("\nChild is Exiting...\n");
}

void parent()
{
printf("\nparent's PID= %d",
getpid());
sleep(20);
printf("\nparent is Exiting...\n");
}
################################

#include<iostream>
#include<sys/wait.h>
#include<unistd.h>
#include<stdlib.h>
```

```
using namespace std;
int main()
{
	int i,n;
	cout<<"Enter no. of child:";
	cin>>n;
	cout<<endl;
	pid_t pid;
	for(i=1;i<=n;i++)
	{
		pid=fork();
	if(pid==0)
	{

		{
			cout<<"I m child"<<endl;
			cout<<"child "<<i<<" id
is:"<<getpid()<<endl;
			cout<<"child "<<i<<" parent id
is:"<<getppid()<<endl;
			break;
		}
	}
	else if(pid>0)
	{
		wait(NULL);
		cout<<"I am Parent"<<endl;
		cout<<"[parent]My id is:"<<getpid()<<endl;
```

```
            cout<<"[parent]parent id
is:"<<getppid()<<endl;

        }
        }
        return 0;
}
```