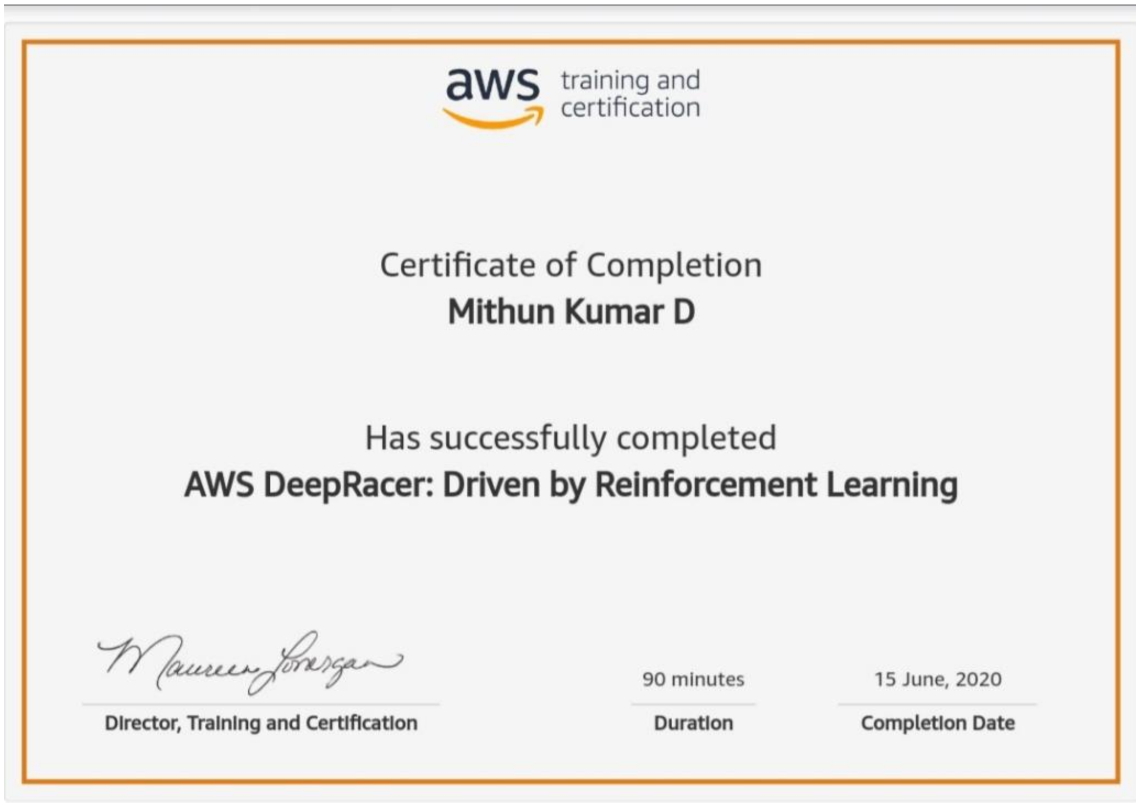
# DAILY ONLINE ACTIVITIES SUMMARY

| | | | |
|---|---|---|---|
| **Date:** | **15/06/2020** | **Name:** | **Mithun Kumar D** |
| **Sem & Sec** | **VIII Semester & A section** | **USN:** | **4AL16CS053** |
| **Online Test Summary** | | | |
| **Subject** | **SMS** | | |
| **Max. Marks** | **60** | **Score** | **No mail received** |
| **Certification Course Summary** | | | |
| **Course** | **AWS DeepRacer: Driven by Reinforcement Learning** | | |
| **Certificate Provider** | **AWS** | **Duration** | **90 minutes** |
| **Coding Challenges** | | | |
| **Problem Statement:** C program to perform operations on triply linked list. | | | |
| **Status: COMPLETED** | | | |
| **Uploaded the report in Github** | **YES** | | |
| **If yes Repository name** | **mkd18** | | |
| **Uploaded the report in slack** | **YES** | | |

**Certification Course Details:**



aws training and certification

Certificate of Completion

**Mithun Kumar D**

Has successfully completed

**AWS DeepRacer: Driven by Reinforcement Learning**

| | | |
|---|---|---|
| [illegible] | 90 minutes | 15 June, 2020 |
| **Director, Training and Certification** | **Duration** | **Completion Date** |

## Coding Challenges Details:

**PROGRAM:** To perform operations on triply linked list.

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


void create(); void
display();
void insert_begin(); void
insert_end(); void
insert_pos(); void
delete_begin(); void
delete_end(); void
delete_pos();


struct node
{
        int info;
        struct node *next;
};
struct node *start=NULL; int main()
{
        int choice;
        while(1){

                printf("\n                    MENU                                  \n");
                printf("\n 1.Create              \n");
                printf("\n 2.Display             \n");
                printf("\n 3.Insert at the beginning                \n");
                printf("\n 4.Insert at the end  \n");
                printf("\n 5.Insert at specified position           \n"); printf("\n
                6.Delete from beginning                             \ n"); printf("\n
                7.Delete from the end                               \n"); printf("\n
                8.Delete from specified position                     n");
```

```
printf("\n 9.Exit                \n");
printf("\n-------------------------------------------------- \n");
printf("\Enter your choice:\t");
scanf("%d",&choice); switch(choice)
{
            case 1:
                        create();
                        break;
            case 2:
                        display(); break;

            case 3:     insert_begin();
                        break;

            case 4:     insert_end(); break;

                        insert_pos(); break;
            case 5:
                        delete_begin();
                        break;
            case 6:
                        delete_end();
                        break;
            case 7:
                        delete_pos();
                        break;
            case 8:



            case 9:
                        exit(0);
                        break;

        default:
```

```
                                        printf("n Wrong Choice:n"); break;
                }
        }
        return 0;
}
void create()
{
        struct node *temp,*ptr;
        temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
        {
                printf("nOut of Memory Space:n"); exit(0);
        }
        printf("nEnter the data value for the node:t");
        scanf("%d",&temp->info);
        temp->next=NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                ptr=start;
                while(ptr->next!=NULL)
                {
                        ptr=ptr->next;
                }
                ptr->next=temp;
        }
}
void display()
{
        struct node *ptr;
        if(start==NULL)
        {
                printf("nList is empty:n"); return;

        }
        else
        {       ptr=start;
                printf("nThe List elements are:n");
                while(ptr!=NULL)
                {
                        printf("%dt",ptr->info );
                        ptr=ptr->next ;
                }
        }
}
void insert_begin()
{
        struct node *temp;
        temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
        {
                printf("nOut of Memory Space:n"); return;
        }
        printf("nEnter the data value for the node:t" );
        scanf("%d",&temp->info);
        temp->next =NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                temp->next=start;
                start=temp;

        }
```

```
}
void insert_end()
{
        struct node *temp,*ptr;
        temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
        {
                printf("nOut of Memory Space:n"); return;
        }
        printf("nEnter the data value for the node:t" );
        scanf("%d",&temp->info );
        temp->next =NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                ptr=start;
                while(ptr->next !=NULL)
                {
                        ptr=ptr->next ;
                }
                ptr->next =temp;
        }
}
void insert_pos()
{
        struct node *ptr,*temp; int i,pos;
        temp=(struct node *)malloc(sizeof(struct node)); if(temp==NULL)
        {
                printf("nOut of Memory Space:n"); return;
        }
        printf("nEnter the position for the new node to be inserted:t"); scanf("%d",&pos);
        printf("nEnter the data value of the node:t");
        scanf("%d",&temp->info) ;

        temp->next=NULL; if(pos==0)
        {
                temp->next=start;
                start=temp;
        }
        else
        {       for(i=0,ptr=start;i<pos-1;i++) { ptr=ptr->next; if(ptr==NULL)
                        {
                                printf("nPosition not found:[Handle with care]n"); return;
                        }
                }
                temp->next =ptr->next ; ptr-
                >next=temp;
        }
}
void delete_begin()
{
        struct node *ptr;
        if(ptr==NULL)
        {
                printf("nList is Empty:n"); return;

        }
        else
        {       ptr=start;
```

```
                start=start->next ;
                printf("nThe deleted element is :%dt",ptr->info); free(ptr);
        }
}
void delete_end()
{
        struct node *temp,*ptr; if(start==NULL)
        {
                printf("nList is Empty:"); exit(0);
        }
        else if(start->next ==NULL)
        {
                ptr=start; start=NULL;
                printf("nThe deleted element is:%dt",ptr->info); free(ptr);


        }
        else    ptr=start;
        {       while(ptr->next!=NULL)
                {
                        temp=ptr; ptr=ptr-
                        >next;
                }
                temp->next=NULL;
                printf("nThe deleted element is:%dt",ptr->info); free(ptr);
        }
}
void delete_pos()
{
        int i,pos;
```

```
          struct node *temp,*ptr; if(start==NULL)
          {
                    printf("nThe List is Empty:n"); exit(0);

          }
          else
          {         printf("nEnter the position of the node to be deleted:t"); scanf("%d",&pos);
                    if(pos==0)
                    {
                              ptr=start; start=start-
                              >next ;
                              printf("nThe deleted element is:%dt",ptr->info ); free(ptr);

                    }
                    else
                    {         ptr=start;
                              for(i=0;i<pos;i++) { temp=ptr; ptr=ptr->next ; if(ptr==NULL)
                                        {
                                                  printf("nPosition not Found:n"); return;
                                        }
                              }
                              temp->next =ptr->next ;
                              printf("nThe deleted element is:%dt",ptr->info ); free(ptr);
                    }
          }
}
```

```
exit(0); break;
default:
```