**const int Mark = 19980310;**

**struct POINT { long long Min, Max, Sum, LazyI, LazyC;**

**} F[MaxN \* 4];**

**void Build (int X, int Y, int Num)**

**{**

**if (X == Y)**

**{ F[Num].Sum = V[X]; F[Num].Min = V[X];**

**F[Num].Max = V[X]; F[Num].LazyC = Mark;**

**return;}**

**int Mid = (X + Y) >> 1;**

**Build(X, Mid, Num \* 2);Build(Mid + 1, Y, Num \* 2 + 1);**

**F[Num].Sum = F[Num \* 2].Sum + F[Num \* 2 + 1].Sum;**

**F[Num].Max = Max(F[Num \* 2].Max, F[Num \* 2 + 1].Max);**

**F[Num].Min = Min(F[Num \* 2].Min, F[Num \* 2 + 1].Min);**

**F[Num].LazyC = Mark;**

**}**

**void MakeLazyC(int Num, int L, int R)**

**{**

**F[Num].Sum = F[Num].LazyC \* (R - L + 1);**

**F[Num].Min = F[Num].LazyC;**

**F[Num].Max = F[Num].LazyC;**

**F[Num].LazyI = 0;**

**if (L != R) F[Num \* 2].LazyC = F[Num].LazyC;**

**if (L != R) F[Num \* 2 + 1].LazyC = F[Num].LazyC;**

**F[Num].LazyC = Mark;**

**}**

**void MakeLazyI(int Num, int L, int R)**

**{**

**int Mid = (L + R) >> 1;**

**if (L != R && F[Num \* 2].LazyC != Mark) MakeLazyC(Num \* 2, L, Mid);**

**if (L != R && F[Num \* 2 + 1].LazyC != Mark) MakeLazyC(Num \* 2 + 1, Mid + 1, R);**

**F[Num].Sum += F[Num].LazyI \* (R - L + 1);**

**F[Num].Min += F[Num].LazyI;**

**F[Num].Max += F[Num].LazyI;**

**if (L != R) F[Num \* 2 + 1].LazyI += F[Num].LazyI;**

**if (L != R) F[Num \* 2].LazyI += F[Num].LazyI;**

**F[Num].LazyI = 0;**

**}**

**long long AskSum(int L, int R, int Num)**

**{**

**if (R < X || Y < L) return 0;**

**if (X <= L && R <= Y && F[Num].LazyC != Mark) return F[Num].LazyC \* (R - L + 1);**

**if (X <= L && R <= Y && F[Num].LazyC == Mark) return F[Num].Sum + F[Num].LazyI \* (R - L + 1);**

**if (F[Num].LazyC != Mark) MakeLazyC(Num, L, R);**

**if (F[Num].LazyI != 0) MakeLazyI(Num, L, R);**

**int Mid = L + R >> 1;**

**return (AskSum(L, Mid, Num \* 2) + AskSum(Mid + 1, R, Num \* 2 + 1));**

**}**

**long long AskMin(int L, int R, int Num)**

**{**

**if (R < X || Y < L) return oo;**

**if (X <= L && R <= Y && F[Num].LazyC != Mark) return F[Num].LazyC;**

**if (X <= L && R <= Y && F[Num].LazyC == Mark) return F[Num].Min + F[Num].LazyI;**

**if (F[Num].LazyC != Mark) MakeLazyC(Num, L , R);**

**if (F[Num].LazyI != 0) MakeLazyI(Num, L, R);**

**int Mid = L + R >> 1;**

**return Min(AskMin(L, Mid, Num \* 2), AskMin(Mid + 1, R, Num \* 2 + 1));**

**}**

**long long AskMax(int L, int R, int Num)**

**{**

**if (R < X || Y < L) return (long long )-oo;**

**if (X <= L && R <= Y && F[Num].LazyC != Mark) return F[Num].LazyC;**

**if (X <= L && R <= Y && F[Num].LazyC == Mark) return F[Num].Max + F[Num].LazyI;**

**if (F[Num].LazyC != Mark) MakeLazyC(Num, L , R);**

**if (F[Num].LazyI != 0) MakeLazyI(Num, L, R);**

**int Mid = L + R >> 1;**

**return Max(AskMax(L, Mid, Num \* 2), AskMax(Mid + 1, R, Num \* 2 + 1));**

**}**

**void UpDate(int Num, int L, int R)**

**{**

**int Mid = (L + R) >> 1;**

**if (F[Num \* 2].LazyC != Mark) MakeLazyC(Num \* 2, L, Mid);**

**if (F[Num \* 2 + 1].LazyC != Mark) MakeLazyC(Num \* 2 + 1, Mid + 1, R);**

**F[Num].Sum = F[Num \* 2].Sum + F[Num \* 2 + 1].Sum + F[Num \* 2].LazyI \* (Mid - L + 1) + F[Num \* 2 + 1].LazyI \* (R - Mid);**

**F[Num].Min = Min(F[Num \* 2].Min + F[Num \* 2].LazyI, F[Num \* 2 + 1].Min + F[Num \* 2 + 1].LazyI);**

**F[Num].Max = Max(F[Num \* 2].Max + F[Num \* 2].LazyI, F[Num \* 2 + 1].Max + F[Num \* 2 + 1].LazyI);**

**}**

**void Increase(int L, int R, int Num)**

**{**

**if (R < X || Y < L) return;**

**if (X <= L && R <= Y)**

**{ if (F[Num].LazyC != Mark) MakeLazyC(Num, L, R);**

**F[Num].LazyI += Z; return; }**

**int Mid = (L + R) >> 1;**

**if (F[Num].LazyC != Mark) MakeLazyC(Num, L , R);**

**if (F[Num].LazyI != 0) MakeLazyI(Num, L, R);**

**Increase(L, Mid, Num \* 2);**

**Increase(Mid + 1, R, Num \* 2 + 1);**

**UpDate(Num, L, R);**

**}**

**void Change(int L, int R, int Num)**

**{**

**if (R < X || Y < L) return;**

**if (X <= L && R <= Y)**

**{ F[Num].LazyC = Z;**

**MakeLazyC(Num, L, R);return;}**

**int Mid = (L + R) >> 1;**

**if (F[Num].LazyC != Mark) MakeLazyC(Num, L , R);**

**if (F[Num].LazyI != 0) MakeLazyI(Num, L, R);**

**Change(L, Mid, Num \* 2);**

**Change(Mid + 1, R, Num \* 2 + 1);**

**UpDate(Num, L, R);**

**}**

**// Main()**

**for (i = 1; i <= N; i++) scanf("%lld", &V[i]);**

**Build(1, N, 1);**

**for (i = 1; i <= M; i++) {**

**scanf("%d%d", &C, &Dir);**

**if (C == 1) scanf("%d%d", &X, &Y);**

**else scanf("%d%d%d", &X, &Y, &Z);**

**if (C == 1 && Dir == 1) printf("%lld\n", AskSum(1, N, 1));**

**if (C == 1 && Dir == 2) printf("%lld\n", AskMin(1, N, 1));**

**if (C == 1 && Dir == 3) printf("%lld\n", AskMax(1, N, 1));**

**if (C == 2 && Dir == 1) Increase(1, N, 1);**

**if (C == 2 && Dir == 2) Change(1, N, 1);**

**//X = 2; Y = 2;**

**}**